Final Assignment

Joel Strouts

24 May 2021

Data set 1: Biomedical Data

This analysis has been performed to identify a new screening procedure for carriers of a rare genetic disorder and to describe that procedure's effectiveness.

Data and Methods

The data consists of 194 samples of which 67 are carriers of the disorder and the remaining 127 are not. Each sample has four measurements: `m1`, `m2`, `m3`, and `m4`. The current industry standard screening procedure uses the measurement `m1` only. In this analysis we investigate alternate approaches, first trying different classifiers on the data as-is then considering

Since the aim of this investigation is to compare the classification efficacy of the `m1` measurement against other methods, we begin by training a classifier using only the labelled `m1` data and then use this model as a baseline for comparison. We then applied the same classification method to each of the alternate measurement features and compared their results to those for `m1`.

We chose to scale the data before performing our analysis to make each feature more individually comparable since we are dealing with only a small number (4) and otherwise they take very different ranges of values.

Having no knowledge about the underlying distributions, we tried four classification algorithms on the single features and compared their results: naive Bayes (NB), linear discriminant analysis (LDA), logistic regression (LR), and random forests (RF). For each model we used the same set of 70:30 training/testing data and leave-one-out cross validation.

Naive Bayes is a count-based method, LDA identifies the best hyperplanes for dividing classes of samples, LR is a transformed variant of linear regression where weights are learned iteratively, and RF models average the classification decisions of many decision trees over different subsets of the sample data (more detailed description provided in the appendix).

We then considered the use of combinations of features for classification based on the most promising results from the previous investigation.

Results

The accuracy of the predictions made by each classifier on the labelled single features (`m1`, `m2`, `m3` & `m4`) are shown in Table 1. These initial results support the established notion that the `m1` feature is a sensible choice for classifying the data, and `m4` performing second best on average. These results seem reasonable given the general distributions of classes for each measurement shown in Figure 1. In this plot we see that for measurements `m2` and `m3` the class distributions overlap more than they do for `m1` and `m4` -making observations from each class harder to distinguish using only these features.

	m1	m2	m3	m4
NB_Accuracy	0.81	0.67	0.72	0.76
LDA_Accuracy	0.76	0.67	0.71	0.79
LR_Accuracy	0.79	0.67	0.69	0.76
RF_Accuracy	0.76	0.55	0.64	0.66

Table 1: Model Classification Accuracy on Test Data as Trained on Pairs of Features

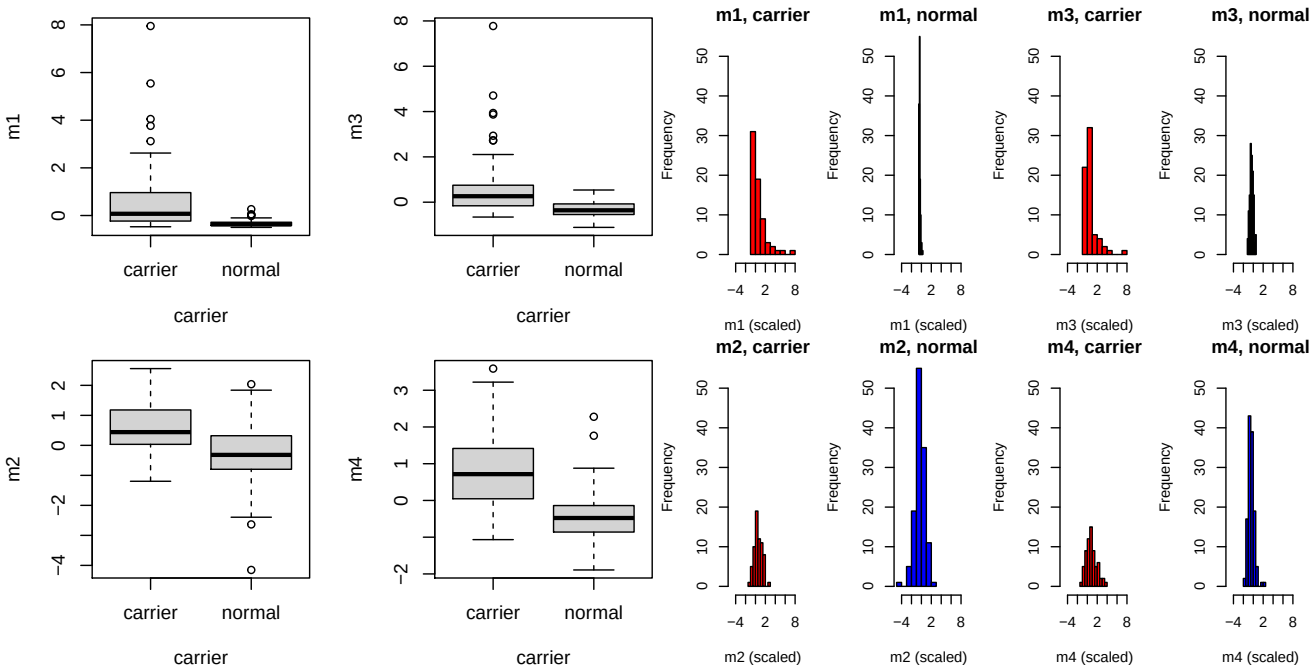

Figure 1: Boxplots and histograms of measurement values for each class

	m1_m2	m1_m3	m1_m4	m2_m3	m2_m4	m3_m4
NB_Accuracy	0.84	0.83	0.84	0.78	0.76	0.76
LDA_Accuracy	0.81	0.71	0.79	0.76	0.76	0.79
LR_Accuracy	0.90	0.78	0.83	0.78	0.76	0.81
RF_Accuracy	0.86	0.76	0.71	0.74	0.76	0.74

Table 2: Model Classification Accuracy on Test Data as Trained on Pairs of Features

Looking at the histogram for the `m4`measurement, a first guess at classifying this data could be to draw a linear decision boundary between the upper quartile of the normal group and the lower quartile of the carrier group, and the fact that the model which identifies a linear decision boundary (LDA) performs best on this measurement supports that intuition somewhat.

Looking at the histograms for each measurement, class distributions can broadly be described as overlapping skew normal, in which case a linear decision boundary would be optimal for just one feature. For a combination of features other models may perform better however.

This is what we investigated next and the results are shown in Table 2. Again we find that `m1` makes the greatest difference to the classification accuracy, with feature pairs including `m1` consistently out-performing the others. The best combination of features seems to be `m1_m2`, with a classification accuracy of `0.9` for the logistic regression model. For the models trained on feature pairs not including `m1` the most accurate was also the logistic regression model, for pair `m3_m4`. Comparing these classification results to the scatter plots in Figure 2 showing the distribution classes for each pair, they seem plausible - the high `m1` measurements for many carriers make those points clearly distinguishable, but all combinations have separate regions dominated by normal samples and dominated by carrier samples respectively, with differing portions of overlap between. In therefore not surprising that the classifiers all have similar (between 0.71 & 0.90) accuracy scores.

After establishing these initial results we investigated possible biases we have reason to suspect exist in the supplied data. We investigate two possible factors: that the age of a person impacts their `m1`-`m4` measurements, and that the measurements made may systematically drift over time. A graph of measurement values (coloured

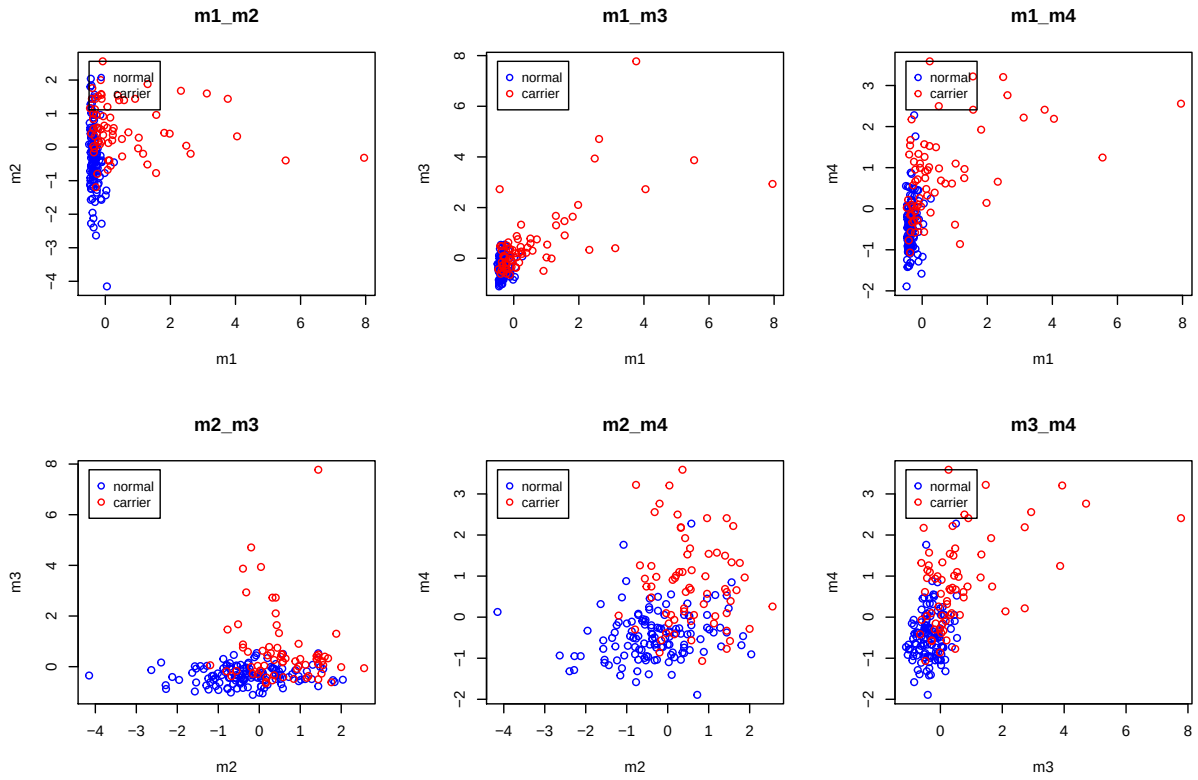

Figure 2: Scatter Plots for Pairs of Features, Coloured by Class

	m1	m2	m3	m4	m1_m2	m1_m3	m1_m4	m2_m3	m2_m4	m3_m4
NB_Accuracy	0.80	0.69	0.82	0.84	0.86	0.88	0.86	0.84	0.84	0.86
LDA_Accuracy	0.82	0.69	0.82	0.84	0.86	0.82	0.86	0.86	0.86	0.84
LR_Accuracy	0.88	0.69	0.82	0.84	0.88	0.88	0.88	0.86	0.86	0.84
RF_Accuracy	0.78	0.73	0.80	0.80	0.88	0.86	0.84	0.86	0.86	0.90

Table 3: Model Classification Accuracy on extitReduced Test Data

according to classification) plotted against age of person tested, and month of measurement (since first measurement recorded) respectively, is shown in Figure 3. Linear trend lines have been imposed on these graphs with 95% confidence intervals to give a sense of the overall relationships between these factors.

Regarding trends, we observe that `m1` and `m2` are generally somewhat higher for younger people, but no clear relationship between age and `m3` and `m4` is discernible. For month of measurement we observe a slight drift towards higher values for `m1` and `m4` over time, and no other clear trends. All of these observations are tentative as the data is few and scattered to draw firm conclusions.

The most striking feature of these graphs, however, is actually the lack of any samples representing people over the age of 40 who are *not* carriers of the condition. For anyone over this age, our predictions are less certain since we do not have representative samples from each class. To investigate the impact this had on our models we ran the same classifiers as before on a subset of the data containing only those ages with samples representing both classes (age < 40). The accuracy of these classifiers trained on a reduced data set are shown in Table 3. The same overall patterns are present: `m1` and `m4` perform better than `m2` and `m3` as single features and pairs of features generally out-perform single features, but in each case the accuracy is an improvement on the models using the entire data set. The model with the highest accuracy is the random forest model trained on the combination of features `m3` and `m4`.

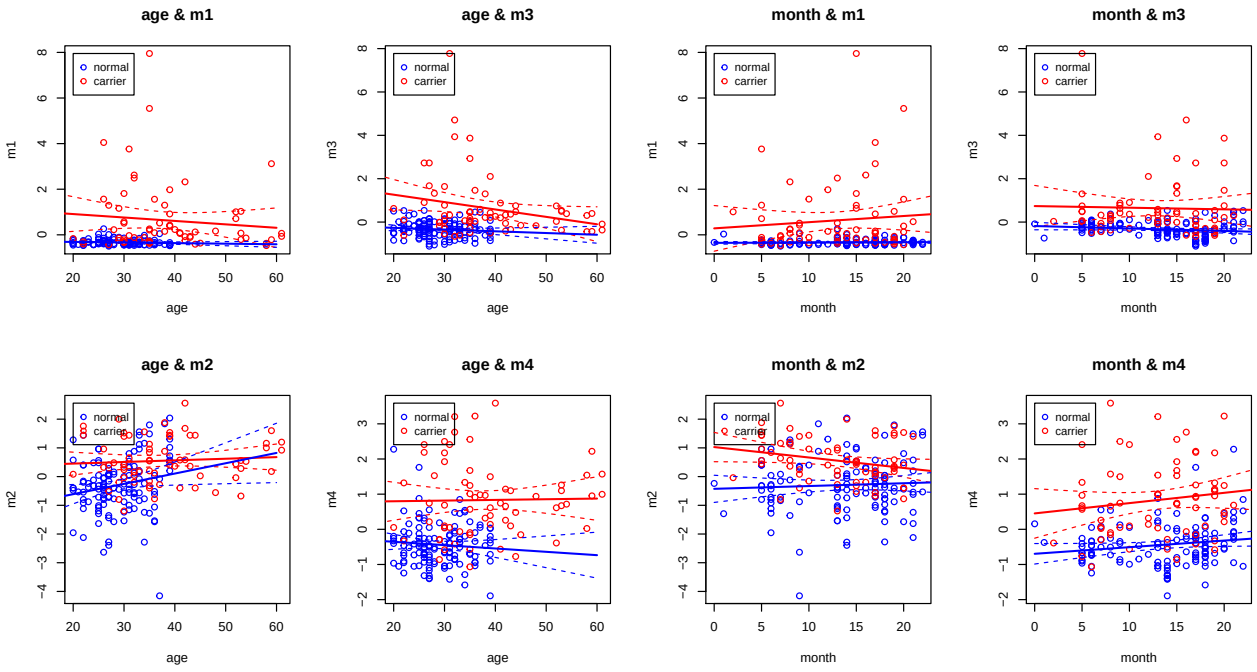

Figure 3: Measurement Values Plotted Against Age & Date Respecitvely

Conclusion

Using the whole data set without considering the effect of age, the best classification accuracy we achieved, using measurements other than `m1`, was `0.81` which is the same as the best classifier we tried on the single feature `m1` for this data. This result came from the logistic regression model trained on features `m3` and `m4`. Using only data which represented both carrier and non-carrier samples in each age group however, all classifiers performed better (including those based on `m1`) with each of naive Bayes', LDA, or logistic regression recording an accuracy of `0.84` based on measurement `m4`. For a combination of features, `m3` and `m4` used to train a random forest model returns an accuracy of `0.90`, the best result recorded.

Based on our analysis we suggest using the `m4` measurement as a replacement for `m1` if a complete replacement is needed or a combination of both if possible and we suggest using a logistic regression model to make these classifications.

Future Work

Regarding the data, it is unbalanced across classes, does not represent each class across a breadth of ages, and the sample size is small. For all of these reasons we suggest the most effective method to increase the accuracy of the screening procedure would be to gather more data.

Regarding the methods investigated in this report, we did not attempt to tune the parameters of any models discussed and did not investigate whether our models were overfitting, which is a concern given the small sample size. If we were to continue our analysis we would choose the best models tested here and tune their parameters and consider the data assumptions of those models in more detail to address overfitting concerns. We also did not investigate the relationships between age & date-of-measurement and recorded value beyond fitting linear trend lines and if these factors were better understood then a classifier which takes them into account could perform better.

Data Set 2: DNA Data

In this analysis we investigate methods for distinguishing between human and bacteriophages DNA sequences. Bacteriophages (or just *phages*) are viruses that infect bacteria and regulate populations in natural ecosystems. Phages invade the human body like other natural environments but it is currently not clear what impact they have on human health.

Data and Methods

The data consists of 300 human DNA sequences and 300 phage DNA sequences. These sequences are each 100 items in length, and each item is one of the four bases: A, T, C, or G.

The approach we took was to first train a random forest classifier on the complete, un-altered sequence data, and then to extract new features derived from those sequences and see if a random forest classifier trained on these new features could out-perform the original model. In both cases we split the data into 75/25 training/testing groups and used 5-fold cross validation on the training sets.

The features we extracted from the original dataset were: the number of times each individual base appeared in a given sequence (how many As, how many Ts etc.), the number of times any pair of bases appeared alongside eachother (how many ATs, ACs, CGs, etc.), and how many times any combination of three appeared in a sequence (AAT, TCC, AGT, etc.). Finally we included features representing the longest consecutive runs of any single base occurring in a given sequence.

After training an RF classifier on these derived features we investigated the features identified by the model as most important for classification.

Results

A PCA (principal component analysis) showing the general spread of the data as described by just the original sequences and compared to the data described by the derived features (counts of single bases, pairs of bases, and strings of three bases in a row) is shown in Figure 4. Clearly these extracted features help distinguish between the classes better than the unaltered sequences alone; for the unaltered sequences the classes overlap almost entirely along PC1 and PC2, whereas for the count based features the classes occupy distinct regions of PC1, PC2 space (though still show a portion of overlap). The results of the random forest classifiers are shown in Table 4.

The RF classifier trained on the count data performed significantly better achieving a classification accuracy of 0.94.

Finally we investigated the variables individually responsible for the most distinguishing between classes by looking at the variable importances identified by the RF model trained on the count data, and these results are shown in Figure 5. This identified the variable `CG_count` ans by far the most significant deciding factor in determining whether a DNA sequence belonged to the human or phage class. In fact, five out of the top six most important variables include a component of counting occurrences of `CG`: `ACG_count`, `GCG_count` etc.

	Accuracy
Unaltered	0.73
Counts	0.94

Table 4: RF Classification Accuracy on Unaltered Data and Count Respectively

Conclusion

We suggest using a random forest classifier trained on the top six most important extracted features shown in Figure 5 to distinguish phage DNA sequences from human ones.

Future Work

We have not tuned any model parameters for the random forest models used, and we also have not tried training a model on solely the most important features identified by the RF model for count features. In

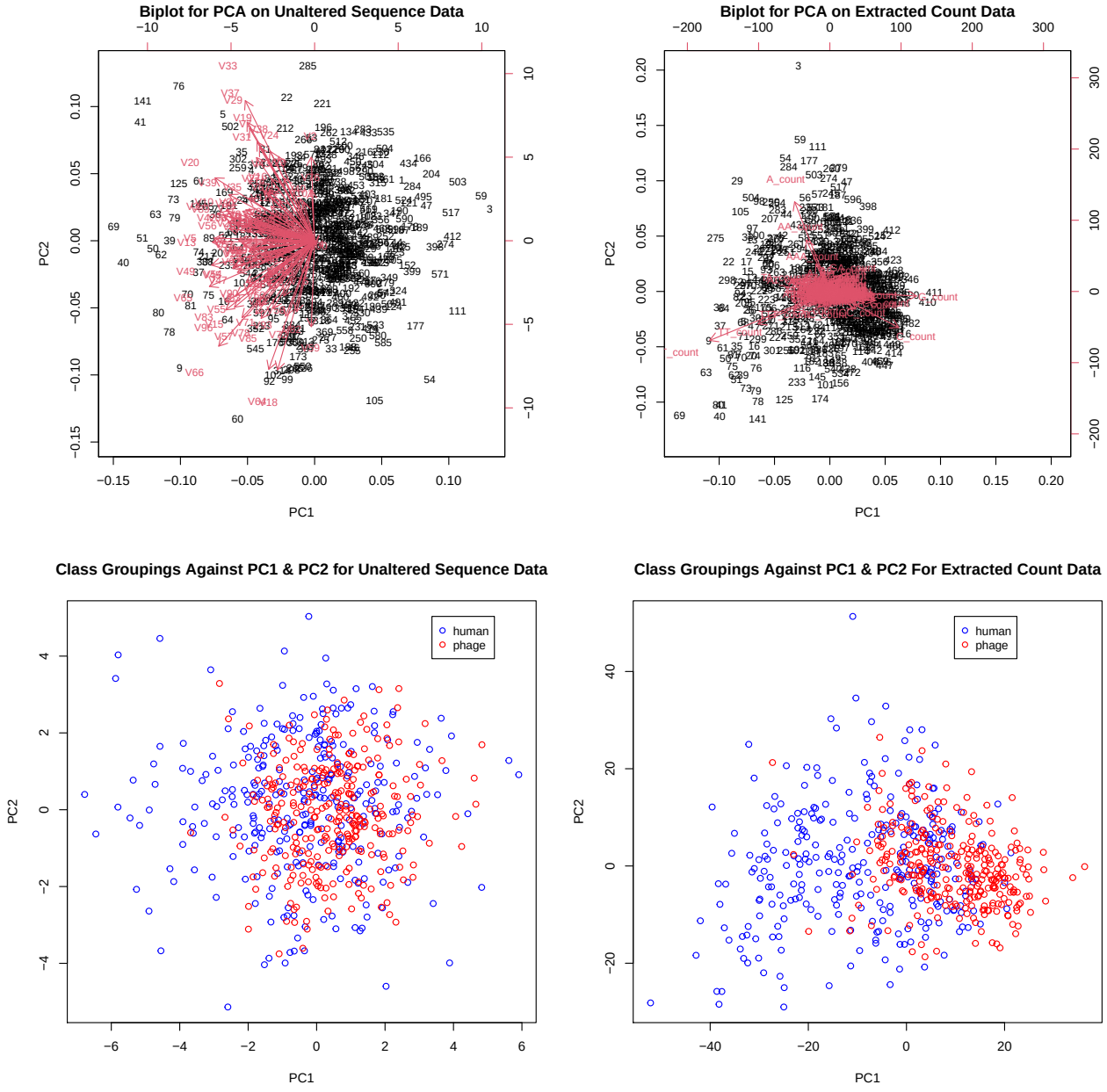

Figure 4: Comparison of Principal Component Analysis on Unaltered Data and Count Data Respectively.

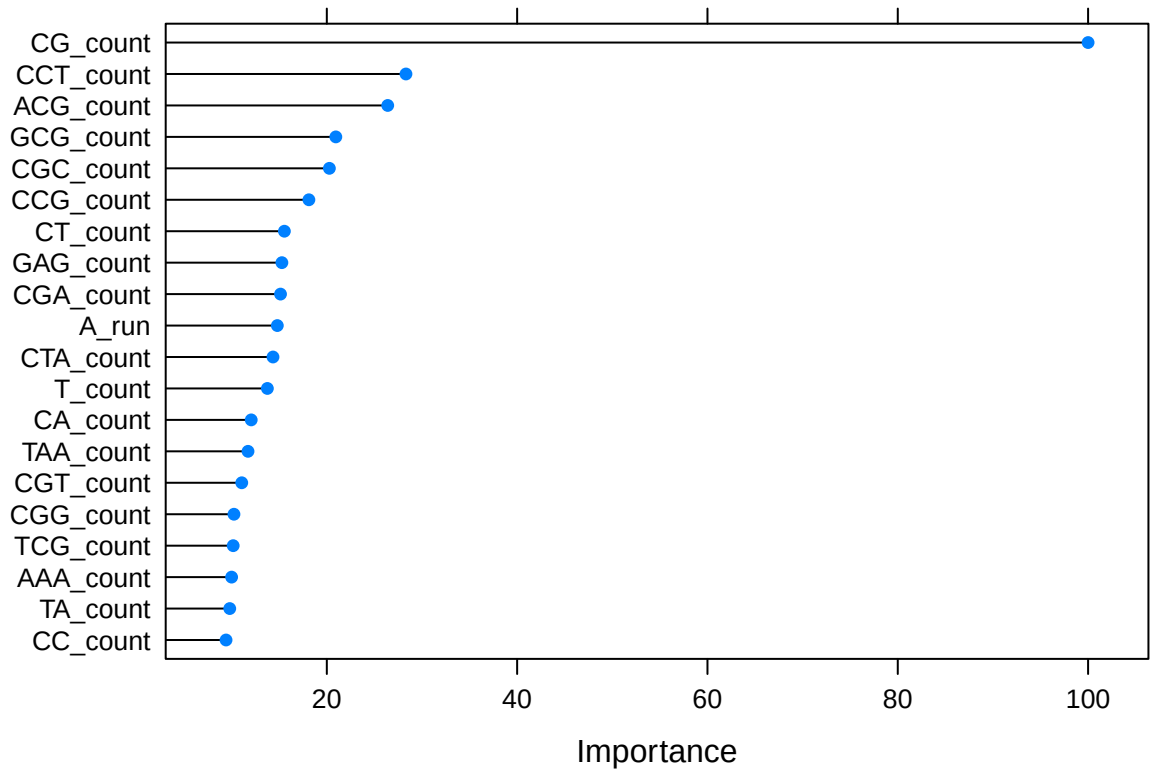

Figure 5: Variable Importance for Derived Count Variables

addition, the 5-fold cross validation we used could be replaced with a more computationally demanding but stable method like leave-one-out cross validation (LOOCV).

Appendix

Naive Bayes

Naive Bayes models work by discretising the data and creating a frequency table which counts how often an object belongs to a given class given it falls within each given range. The conditional probability that an object belongs to a class given it's feature values is then calculated with Bayes' formula using this frequency data, and the class with the highest conditional probability is selected.

Linear Discriminant Analysis

LDA works by empirically identifying the hyperplanes which best divide the data into separate classes.

Logistic Regression

Binary Logistic Regression models work similarly to standard linear regression models except that the terms are transformed by a logistic function, the weights are fitted using gradient descent rather than being explicitly solved for, and they are used for binary classification rather than approximation.

Random Forests

Random forest models work by bootstrapping the data and associating a decision tree with each bootstrap which chooses the of that subset of data. The final classification is then made by aggregating the votes of all the individual decision trees and choosing the class with the most votes.

Code

```
## ------------------------------------------------------ ##
## complete source code for this document can be found at: ##
## https://github.com/joelstrouts/PDS-final-assignment     ##
## ------------------------------------------------------ ##

set.seed(20)
do_re_eval = FALSE

## ===================== ##
## SECTION 1: Blood data ##
## ===================== ##

## ------------------------ ##
## SUBSECTION: preprocessing ##
## ------------------------ ##

models <- list(c("nb", "nb"), c("lda", "lda"),
               c("regLogistic", "lr"), c("rf", "rf"))

## Reading Data
blood <- list()
blood[["dfs"]] <- list()

blood$dfs[["normal"]] <-
    read.table(std$get_fpath("normals.txt"), header = TRUE)
blood$dfs[["carrier"]] <-
    read.table(std$get_fpath("carriers.txt"), header = TRUE)

## Combine data frames into single df with categorical variable distinguishing
## between types
blood[["meta"]] <- list()
blood$meta[["idvs"]] <- c("m1", "m2", "m3", "m4")

blood$dfs$normal$carrier <- rep("normal", nrow(blood$dfs$normal))
blood$dfs$carrier$carrier <- rep("carrier", nrow(blood$dfs$carrier))

blood$dfs$main <- rbind(blood$dfs$normal, blood$dfs$carrier)

blood$dfs$main$date <- str_pad(blood$dfs$main$date, width=6, side="left", pad="0")
blood$dfs$main$date <- as.Date(sub("(..)(..)(..)", "19\\3-\\1-01", blood$dfs$main$date))
blood$dfs$main$month <-
    as.numeric(round(difftime(blood$dfs$main$date, "1978-01-01", unit="days")/30.42))

blood$dfs[["unscaled"]] <- blood$dfs$main
blood$dfs$main[blood$meta$idvs] <-
    scale(blood$dfs$main[blood$meta$idvs])

blood$meta[["age_max"]] <-
    max(blood$dfs$main[which(blood$dfs$main$carrier == "normal"),"age"])
blood$meta[["mixed_diagnosis_ages"]] <-
    which(blood$dfs$main$age <= blood$meta$age_max)
```

```
## END-SECTION: preprocessing ##
## ------------------------- ##

## ------------------ ##
## SUBSECTION: Models ##
## ------------------ ##

## partition data
blood$meta[["train_idxs"]] <- createDataPartition(blood$dfs$main$carrier, p=0.7, list=FALSE)
blood$dfs[["train"]] <- blood$dfs$main[blood$meta$train_idxs,]
blood$dfs[["test"]] <- blood$dfs$main[-blood$meta$train_idxs,]

## train models
blood[["mdls"]] <- list()
blood$mdls[["pca-unscaled"]] <- prcomp(blood$dfs$unscaled[blood$meta$idvs])
blood$mdls[["pca-scaled"]] <- prcomp(blood$dfs$main[blood$meta$idvs])

# single feature classification
if(do_re_eval) {

    for(model in models) {
        blood$mdls[[model[[2]]]] <-
            std$multi_train(blood$dfs$train, blood$dfs$test,
                            blood$meta$idvs, "carrier",
                            model[[1]], trainControl(method = "LOOCV"))
    }
    store_blood_mdls <- blood$mdls
    save(store_blood_mdls, file='store/blood_mdls.Rda')

} else {

    load("store/blood_mdls.Rda")
    blood$mdls <- store_blood_mdls

}

blood$mdls[["ft_pairs"]] <- list()
blood$meta[["var_pairs"]] <- list(
    c("m1", "m2"), c("m1", "m3"), c("m1", "m4"),
    c("m2", "m3"), c("m2", "m4"), c("m3", "m4"))

# feature pair models
if(do_re_eval) {

    for(model in models) {
        blood$mdls$ft_pairs[[model[[2]]]] <-
            std$multi_train(
                    blood$dfs$train, blood$dfs$test, blood$meta$var_pairs,
                    "carrier", model[[1]], trainControl(method = "LOOCV"))
    }
    store_blood_mdls_ft_pairs <- blood$mdls$ft_pairs
    save(store_blood_mdls_ft_pairs, file='store/blood_mdls_ft_pairs.Rda')
```

```
} else {

    load("store/blood_mdls_ft_pairs.Rda")
    blood$mdls$ft_pairs <- store_blood_mdls_ft_pairs

}

## partition data
blood$dfs[["reduced"]] <- blood$dfs$main[blood$meta$mixed_diagnosis_ages,]
blood$meta[["train_idxs_red"]] <- createDataPartition(blood$dfs$reduced[,1], p=0.7, list=FALSE)

blood$dfs[["train_red"]] <- blood$dfs$reduced[blood$meta$train_idxs_red,]
blood$dfs[["test_red"]] <- blood$dfs$reduced[-blood$meta$train_idxs_red,]

blood$mdls[["red"]] <- list()

## single feature classification on reduced data
if(do_re_eval) {

    for(model in models) {
        blood$mdls$red[[model[[2]]]] <-
            std$multi_train(
                    blood$dfs$train_red, blood$dfs$test_red, blood$meta$idvs,
                    "carrier", model[[1]], trainControl(method = "LOOCV"))
    }
    store_blood_mdls_red <- blood$mdls$red
    save(store_blood_mdls_red, file='store/blood_mdls_red.Rda')

} else {

    load("store/blood_mdls_red.Rda")
    blood$mdls$red <- store_blood_mdls_red

}

blood$mdls$red[["ft_pairs"]] <- list()
# feature pair models on reducced data
if(do_re_eval) {

    for(model in models) {
        blood$mdls$red$ft_pairs[[model[[2]]]] <-
            std$multi_train(
                    blood$dfs$train_red, blood$dfs$test_red, blood$meta$var_pairs,
                    "carrier", model[[1]], trainControl(method = "LOOCV"))
    }
    store_blood_mdls_red_ft_pairs <- blood$mdls$red$ft_pairs
    save(store_blood_mdls_red_ft_pairs, file='store/blood_mdls_red_ft_pairs.Rda')

} else {

    load("store/blood_mdls_red_ft_pairs.Rda")
    blood$mdls$red$ft_pairs <- store_blood_mdls_red_ft_pairs
```

```
}

## END-SECTION: Models ##
## ------------------- ##

## ------------------- ##
## SUBSECTION:  tables ##
## ------------------- ##

blood[["tables"]] <- list()

blood$tables[["model_results_overview"]] <- function(model_name, model_set) {
    model <- model_set[[model_name]]
    first_var <- model$id_vars[[1]]
    # initialize table with just first column
    comparison_table <- data.frame(
        as.data.frame(model$results[[first_var]]$overall)[,1])
    rename_candidate <- names(comparison_table)[[1]]
    names(comparison_table)[
        names(comparison_table) == rename_candidate] <- first_var
    # rename rows
    new_row_names <- row.names(as.data.frame(model$results[[first_var]]$overall))
    new_row_names <- lapply(new_row_names,
               function(metric) {paste(toupper(model_name), metric, sep="_")})
    row.names(comparison_table) <- new_row_names
    # add other rows
    for(var in model$id_vars[-1]) {
        comparison_table[[var]] <-
            as.data.frame(model$results[[var]]$overall)[,1]
    }
    comparison_table
}

blood$tables[["compare_model_accuracies"]] <- function(model_names, model_set) {
    table <- blood$tables$model_results_overview(model_names[[1]], model_set)[1,]
    for(model_name in model_names[-1]) {
        table <- rbind(
            table,
            blood$tables$model_results_overview(model_name, model_set)[1,]
            )
    }
    table
}

## END-SECTION: tables ##
## ------------------- ##

## ------------------- ##
## SUBSECTION:  plots ##
## ------------------- ##

blood[["plots"]] <- list()
```

```
blood$plots[["class_histograms"]] <- function(variant) {
    layout(matrix(c(1,3,2,4,5,7,6,8), ncol=4, nrow=2))
    for(var in c("m1", "m2", "m3", "m4")) {
        for(class in c("carrier", "normal")) {
            if(class == "normal") {
                hist_col <- "blue"
            } else {
                hist_col <- "red"
            }
            hist_data <-
                blood$dfs$main[which(blood$dfs$main$carrier == class),var]
            if(variant == "auto-scale") {
                hist(hist_data, col=hist_col,
                     main=paste(var, class, sep = ", "),
                     xlab = paste(var, "(scaled)"))
            } else if(variant == "same-scale") {
                hist(hist_data, col=hist_col,
                     main=paste(var, class, sep = ", "),
                     xlab = paste(var, "(scaled)"),
                     xlim = c(-5,8), ylim=c(0,55))
            }
        }
    }
}

blood$plots[["class_qqnorms"]] <- function(variant) {
    layout(matrix(c(1,3,2,4,5,7,6,8), ncol=4, nrow=2))
    for(var in c("m1", "m2", "m3", "m4")) {
        for(class in c("normal", "carrier")) {
            if(class == "normal") {
                hist_col <- "blue"
            } else {
                hist_col <- "red"
            }
            hist_data <-
                blood$dfs$main[which(blood$dfs$main$carrier == class),var]
            qqnorm(hist_data, col=hist_col,
                 main=paste(var, class, sep = ", "), xlab = var)
        }
    }
}

blood$plots[["boxplots"]] <- function() {
    layout(matrix(1:4, nrow=2, ncol=2))
    boxplot(m1 ~ carrier, data=blood$dfs$main)
    boxplot(m2 ~ carrier, data=blood$dfs$main)
    boxplot(m3 ~ carrier, data=blood$dfs$main)
    boxplot(m4 ~ carrier, data=blood$dfs$main)
}

blood$plots[["pca"]] <- function() {
    layout(matrix(1:1, ncol=2, nrow=2))
    biplot(pca_blood)
```

```
    biplot(pca_blood_scaled)
    std$my_scatter__col_levels(
        pca_blood$x[,1], pca_blood$x[,2],
        color_by = blood$dfs$main$carrier,
        labels = TRUE,
        legend_pos = c(0.7,0.7))
    std$my_scatter__col_levels(
        pca_blood_scaled$x[,1], pca_blood_scaled$x[,2],
        color_by = blood$dfs$main$carrier,
        labels = TRUE,
        legend_pos = c(0.7,0))
}

blood$plots[["month_age_scatters"]] <- function() {
    layout(matrix(1:8, nrow=2, ncol=4))
    for(idv in c("age", "month")) {
        for(dv in c("m1","m2","m3","m4")) {
            std$my_scatter__col_levels(
                blood$dfs$main[,idv],
                blood$dfs$main[,dv],
                axis_labs = c(idv, dv),
                color_by = blood$dfs$main$carrier,
                colors = c("red", "blue"),
                plot_title = paste0(idv, " & ", dv))
            for(class in c("carrier", "normal")) {
                df <- blood$dfs$main[which(
                                blood$dfs$main$carrier==class),]
                fit <- lm(
                    as.formula(paste0(
                        dv, "~poly(", idv, ",1)"
                        )),
                    data=df)
                idv_range <-
                    seq(0, 60, length.out = 140)
                newdat <- data.frame(idv = idv_range)
                names(newdat) <- c(idv)
                color <- ifelse((class == "carrier"), "red", "blue")
                pred <- predict(fit, newdata = newdat, interval = "confidence", level = 0.95)
                lines(x = idv_range, y = pred[,1], col = color, lwd = 1.7)
                lines(x = idv_range, y = pred[,2], col = color,
                      lty = "dashed")
                lines(x = idv_range, y = pred[,3], col = color,
                      lty = "dashed")
            }
        }
    }
}

blood$plots[["pair_scatters"]] <- function() {
    layout(matrix(c(1,4,2,5,3,6), ncol=3, nrow=2))
    for(pair in blood$meta$var_pairs) {
        std$my_scatter__col_levels(
                blood$dfs$main[,pair[[1]]], blood$dfs$main[,pair[[2]]],
```

```
                axis_labs = c(pair[[1]], pair[[2]]),
                plot_title = paste0(pair, collapse="_"),
                color_by = blood$dfs$main$carrier,
                shapes = c(1),
                colors = c("red", "blue"),
                legend_pos = c(0,0))
    }
}

## END-SECTION: plots ##
## ------------------ ##

## END SECTION 1         ##
## ===================== ##


## ================== ##
## SECTION 2: DNA data ##
## ================== ##

## ------------------------- ##
## SUBSECTION:  preprocessing ##
## ------------------------- ##

dna <- list()
dna[["dfs"]] <- list()

dna$dfs[["seqs"]] <- read.table(std$get_fpath("human-phage.txt"))
dna$dfs[["seqs_numeric"]] <- dna$dfs[["seqs"]]

# convert original table to numeric version
for(i in 2:101) {
    ## A->1, C->2 , G->3, T->4
    dna$dfs$seqs_numeric[,i] <-
        as.numeric(as.factor(dna$dfs$seqs[,i]))
}

dna$dfs[["ngrams"]] <- data.frame()
dna$meta[["the2grams"]] <- c()
dna$meta[["the3grams"]] <- c()

if(do_re_eval) {
    for(row in 1:length(dna$dfs$seqs[,1])) {
        for(win_size in 2:3) {
            for(col in 2:(101 - win_size)) {
                bases <- ""
                for(offset in 1:win_size) {
                    bases <- paste(bases, dna$dfs$seqs[row,col-1+offset], sep = "")
                }
                dna$dfs$ngrams[row,paste(col, win_size, sep = "_")] <- bases
                if(win_size == 2) {
                    dna$meta$the2grams <- c(dna$meta$the2grams, bases)
                } else if(win_size == 3) {
```

```
                    dna$meta$the3grams <- c(dna$meta$the3grams, bases)
                }
            }
        }
    }
    store_dna_ngrams <- list(
        ngrams_df = dna$dfs$ngrams,
        the2grams_v = dna$meta$the2grams,
        the3grams_v = dna$meta$the3grams
    )
    save(store_dna_ngrams, file="store/dna_ngrams.Rda")
} else {
    load("store/dna_ngrams.Rda")
    dna$dfs[["ngrams"]] <- store_dna_ngrams$ngrams_df
    dna$meta[["the2grams"]] <- store_dna_ngrams$the2grams_v
    dna$meta[["the3grams"]] <- store_dna_ngrams$the3grams_v
}

dna$dfs[["the2grams"]] <-
    data.frame(seq = unique(as.factor(dna$meta$the2grams)))
dna$dfs[["the3grams"]] <-
    data.frame(seq = unique(as.factor(dna$meta$the3grams)))

for(row in 1:length(dna$dfs$the2grams[,1])) {
    dna$dfs$the2grams[row,"count"] <-
        length(which(dna$meta$the2grams == dna$dfs$the2grams[row,"seq"]))
}

for(row in 1:length(dna$dfs$the3grams[,1])) {
    dna$dfs$the3grams[row,"count"] <-
        length(which(dna$meta$the3grams == dna$dfs$the3grams[row,"seq"]))
}

if(do_re_eval) {
    dna$dfs[["counts"]] <- data.frame(dna$dfs$seqs$V1)

    names(dna$dfs$counts)[names(dna$dfs$counts)=="dna.dfs.seqs.V1"] <- "class"

    for(row in 1:length(dna$dfs$seqs[,1])) {
        for(base in c("A", "C", "G", "T")) {
            longest_run <- 0
            current_run <- 0
            for(col in 1:length(dna$dfs$seqs[1,])) {
                if(dna$dfs$seqs[row, col] == base) {
                    current_run <- current_run + 1
                } else {
                    if(current_run > longest_run) {
                        longest_run <- current_run
                    }
                    current_run <- 0
                }
            }
            dna$dfs$counts[row, paste(base, "run", sep="_")] <- longest_run
```

```
        }
    }

    for(row in 1:length(dna$dfs$seqs[,1])) {
        print(paste("doing row", row))
        for(base in c("A", "C", "G", "T")) {
            dna$dfs$counts[row, paste0(base, "_count")] <-
                length(which(dna$dfs$seqs[row,2:101] == base))
        }
        for(seq in dna$dfs$the2grams[,"seq"]) {
            dna$dfs$counts[row, paste0(seq, "_count")] <-
                length(which(as.factor(dna$dfs$ngrams[row,]) == seq))
        }
        for(seq in dna$dfs$the3grams[,"seq"]) {
            dna$dfs$counts[row, paste0(seq, "_count")] <-
                length(which(as.factor(dna$dfs$ngrams[row,]) == seq))
        }
    }
    store_dna_counts <- dna$dfs$counts
    save(store_dna_counts, file="store/dna_counts.Rda")
} else {

    load("store/dna_counts.Rda")
    dna$dfs$counts <- store_dna_counts

}


for(i in 1:600) {
    if(dna$dfs$seqs_numeric[i,1] == "pos") {
        dna$dfs$seqs_numeric[i,1] <- "human"
        dna$dfs$seqs[i,1] <- "human"
        dna$dfs$counts[i,1] <- "human"
    } else if(dna$dfs$seqs_numeric[i,1] == "neg") {
        dna$dfs$seqs_numeric[i,1] <- "phage"
        dna$dfs$seqs[i,1] <- "phage"
        dna$dfs$counts[i,1] <- "phage"
    }
}

## END-SECTION: preprocessing ##
## -------------------------- ##

## ------------------- ##
## SUBSECTION:  Models ##
## ------------------- ##

dna[["mdls"]] <- list()

dna$meta[["train_idxs"]] <-
    createDataPartition(dna$dfs$seqs_numeric[,1], p=0.75, list=FALSE)

dna$dfs[["seqs_train"]] <- dna$dfs$seqs_numeric[dna$meta$train_idxs,]
```

```
dna$dfs[["seqs_test"]] <- dna$dfs$seqs_numeric[-dna$meta$train_idxs,]

dna$dfs[["counts_train"]] <- dna$dfs$counts[dna$meta$train_idxs,]
dna$dfs[["counts_test"]] <- dna$dfs$counts[-dna$meta$train_idxs,]

dna$mdls[["seqs_pca"]] <- prcomp(dna$dfs$seqs_numeric[,-1])
dna$mdls[["counts_pca"]] <- prcomp(dna$dfs$counts[,-1])

if(do_re_eval) {

    dna$mdls[["seqs_rf"]] <-
        std$train_model(
                dna$dfs$seqs_train, dna$dfs$seqs_test,
                -1, "V1", "rf", trainControl(method = "cv", number = 5), import = TRUE)

    dna$mdls[["counts_rf"]] <-
        std$train_model(
                dna$dfs$counts_train, dna$dfs$counts_test,
                -1, "class", "rf", trainControl(method = "cv", number = 5), import = TRUE)

    store_dna_mdls <- dna$mdls
    save(store_dna_mdls, file = "store/dna_mdls.Rda")

} else {
    load("store/dna_mdls.Rda")
    dna$mdls <- store_dna_mdls
}

## END-SECTION: Models ##
## ------------------- ##

## ------------------- ##
## SUBSECTION:  tables ##
## ------------------- ##

dna[["tables"]] <- list()

dna$tables[["rf_compare"]] <- function() {
    table <- rbind(
        dna$mdls$seqs_rf$results$overall,
        dna$mdls$counts_rf$results$overall
    )
    row.names(table) <- c("Unaltered", "Counts")
    names(table) <- c("Accuracy")
    data.frame(Accuracy = table[,c(1)])
}

## END-SECTION: tables ##
## ------------------- ##

## ------------------ ##
## SUBSECTION:  plots ##
## ------------------ ##
```

```
dna[["plots"]] <- list()

dna$plots[["pca_compare"]] <- function() {
    layout(matrix(1:4, nrow=2, ncol=2))
    biplot(dna$mdls$seqs_pca, main="Biplot for PCA on Unaltered Sequence Data")
    std$my_scatter__col_levels(
        dna$mdls$seqs_pca$x[,1], dna$mdls$seqs_pca$x[,2],
        color_by = dna$dfs$seqs[,1],
        colors = c("blue", "red"),
        axis_labs = c("PC1", "PC2"),
        plot_title = "Class Groupings Against PC1 & PC2 for Unaltered Sequence Data",
        legend_pos = c(0.8,0))
    biplot(dna$mdls$counts_pca, main="Biplot for PCA on Extracted Count Data")
    std$my_scatter__col_levels(
        dna$mdls$counts_pca$x[,1], dna$mdls$counts_pca$x[,2],
        color_by = dna$dfs$counts[,1],
        colors = c("blue", "red"),
        axis_labs = c("PC1", "PC2"),
        plot_title = "Class Groupings Against PC1 & PC2 For Extracted Count Data",
        legend_pos = c(0.7,0))
}

dna$plots[["rf_importance"]] <- function() {
    rf_var_imp <- varImp(dna$mdls$counts_rf$mdl)
    plot(rf_var_imp, top = 20)
}

## END-SECTION: plots ##
## ------------------ ##

initialised_workspace <- TRUE
```