IDS Final Project

Group Members: Dan Hu, Sam Anderson, Ken Shelley

West Section

```
# Load all the libraries we need.
library(tidyverse)
library(stringr)
library(rpart)
library(partykit)
library(randomForest)
library(class)
```

```
# Load the data set. Assign it to the variable 'fna'.
fna <- read_csv("FNA_cancer.csv")
glimpse(fna)
```

```
# Make a copy of the original data set and delete the first and last column which are useless. Assign it with a n
ew variable 'fna_new'.
fna_new <- data.frame(fna)
fna_new <- fna_new %>% dplyr::select(-id, -X33)
glimpse(fna_new)
```

```
# Replace the dot sign in column names with underscore so that all column names are in uniform form.
names(fna_new) <- str_replace_all(names(fna_new), '[.]', '_')

# Conver the response variable 'diagnosis' into factor.
fna_new$diagnosis <- as.factor(fna_new$diagnosis)

glimpse(fna_new)
```

2. Perform basic exploratory data analysis.

```
# Display the numerical summary of each characteristic in the data.
summary(fna_new)
```

```
# Create histograms for each characteristic to see the distributions of their values.
characteristics <- names(fna_new)[-1]
for (i in 1:30){
  p <- ggplot(fna_new, aes(x = get(characteristics[i]), fill = diagnosis)) +
    geom_histogram() +
    xlab(characteristics[i])
  print(p)
}
```

3. Split the data into test and training data.

```
# Set the seed to 1847
set.seed(1847)

# Get the total rows of the dataset. Assign the result to the variable 'n'.
n <- nrow(fna_new)

# Randomly generate the indexes for the test set. Use the indexes to select the 20% test set. Assign the test set
to the variable 'test'.
test_idx <- sample.int(n, size = round(0.2 * n))
test <- fna_new[test_idx, ]

# Select the rest 80% training data and assign it to the variable 'train'.
train <- fna_new[-test_idx, ]

# Give a glimpse of the resulting training set.
glimpse(train)

# Give a glimpse of the resulting testing set.
glimpse(test)
```

4. Build a classification algorithm using decision trees. Prune your tree appropriately.

```
# Create the regression formula, take 'diagnosis' as the response variable and all the rest 30 characteristics as
the predictor variables. Assign the formula to the variable 'form'.
form <- as.formula(diagnosis ~ .)

# Create a tree using the formula just created. Assign the new tree to the variable 'diagnosis_tree'.
diagnosis_tree <- rpart(form, data = train)
plot(as.party(diagnosis_tree))

# Use 'diagnosis_tree' model to predict the test data and assign the predictions to the variable 'test_pred_tre
e'.
test_pred_tree <- predict(diagnosis_tree, newdata = test, type = 'class')

# Create the confusion matrix for the test data set. Assign the result to the variable 'confusion_tree'.
confusion_tree <- table(test$diagnosis, test_pred_tree)
confusion_tree
```

```
printcp(diagnosis_tree)
```

```
# Create the regression formula, take 'diagnosis' as the response variable and the 10 mean values of the characte
ristics as the predictor variables. Assign the formula to the variable 'form_mean'.
form_mean <- as.formula("diagnosis ~ radius_mean + texture_mean + perimeter_mean + area_mean + smoothness_mean +
 compactness_mean + concavity_mean + concave_points_mean + symmetry_mean + fractal_dimension_mean")

# Create a tree using the formula just created. Assign the new tree to the variable 'diagnosis_tree_mean'.
diagnosis_tree_mean <- rpart(form_mean, data = train)
plot(as.party(diagnosis_tree_mean))

# Use 'diagnosis_tree_mean' model to predict the test data and assign the predictions to the variable 'test_pred_
tree_mean'.
test_pred_tree_mean <- predict(diagnosis_tree_mean, newdata = test, type = 'class')

# Create the confusion matrix for the test data set. Assign the result to the variable 'confusion_tree_mean'.
confusion_tree_mean <- table(test$diagnosis, test_pred_tree_mean)
confusion_tree_mean
```

```
printcp(diagnosis_tree_mean)
```

```
# Create the regression formula, take 'diagnosis' as the response variable and the 10 standard error of the mean
 of the characteristics as the predictor variables. Assign the formula to the variable 'form_se'.
form_se <- as.formula("diagnosis ~ radius_se + texture_se + perimeter_se + area_se + smoothness_se + compactness_
se + concavity_se + concave_points_se + symmetry_se + fractal_dimension_se")

# Create a tree using the formula just created. Assign the new tree to the variable 'diagnosis_tree_se'.
diagnosis_tree_se <- rpart(form_se, data = train)
plot(as.party(diagnosis_tree_se))

# Use 'diagnosis_tree_se' model to predict the test data and assign the predictions to the variable 'test_pred_tr
ee_sd'.
test_pred_tree_sd <- predict(diagnosis_tree_se, newdata = test, type = 'class')

# Create the confusion matrix for the test data set. Assign the result to the variable 'confusion_tree_sd'.
confusion_tree_sd <- table(test$diagnosis, test_pred_tree_sd)
confusion_tree_sd
```

```
printcp(diagnosis_tree_se)
```

```
# Create the regression formula, take 'diagnosis' as the response variable and the 10 maximum value of the charac
teristics as the predictor variables. Assign the formula to the variable 'form_worst'.
form_worst <- as.formula("diagnosis ~ radius_worst + texture_worst + perimeter_worst + area_worst + smoothness_wo
rst + compactness_worst + concavity_worst + concave_points_worst + symmetry_worst + fractal_dimension_worst")

# Create a tree using the formula just created. Assign the new tree to the variable 'diagnosis_tree_worst'.
diagnosis_tree_worst <- rpart(form_worst, data = train)
plot(as.party(diagnosis_tree_worst))

# Use 'diagnosis_tree_worst' model to predict the test data and assign the predictions to the variable 'test_pred
_tree_worst'.
test_pred_tree_worst <- predict(diagnosis_tree_worst, newdata = test, type = 'class')

# Create the confusion matrix for the test data set. Assign the result to the variable 'confusion_tree_worst'.
confusion_tree_worst <- table(test$diagnosis, test_pred_tree_worst)
confusion_tree_worst
```

```
printcp(diagnosis_tree_worst)
```

5. Build a classification algorithm using random forests/bagging. Adjust the parameters of the forest appropriately.

```
# Create a bagged(so mtry = 30) set of trees from the training data set. Assign the result to the variable 'diagn
osis_bagging'.
diagnosis_bagging <- randomForest(form, data = train, mtry = 30, ntree = 500, na.action = na.roughfix)
diagnosis_bagging

# Use 'diagnosis_bagging' model to predict the test data and assign the predictions to the variable 'test_pred_ba
gging'.
test_pred_bagging <- predict(diagnosis_bagging, newdata = test, type = 'class')

# Create the confusion matrix for the test data set. Assign the result to the variable 'confusion_bagging'.
confusion_bagging <- table(test$diagnosis, test_pred_bagging)
confusion_bagging
```

```
# Create a random forest from the training data set, set the mtry = 5. Assign the result to the variable 'diagnos
is_forest'.
diagnosis_forest <- randomForest(form, data = train, mtry = 5, ntree = 500, na.action = na.roughfix)
diagnosis_forest

# Use 'diagnosis_forest' model to predict the test data and assign the predictions to the variable 'test_pred_for
est'.
test_pred_forest <- predict(diagnosis_forest, newdata = test, type = 'class')

# Create the confusion matrix for the test data set. Assign the result to the variable 'confusion_forest
confusion_forest <- table(test$diagnosis, test_pred_forest)
confusion_forest
```

6. Build a classification algorithm using Kth Nearest Neighbors. Tune the value of K appropriately.

```
# Create a rescale function, name it 'rescale_x'.
rescale_x <- function(x){(x - min(x)) / (max(x) - min(x))}

# Rescale the training data. Assign the rescaled test data to the cariable 'new_train'.
new_train <- train
new_train[-1] <- map_df(train[-1], rescale_x)
glimpse(new_train)

# Create a rescale function, name it 'rescale_xy'.
rescale_xy <- function(x, y){(x - min(y)) / (max(y) - min(y))}
# Rescale the test data. Assign the rescaled test data to the cariable 'new_test'.
new_test <- test
new_test[-1] <- map2_df(test[-1], train[-1], rescale_xy)
glimpse(new_test)
```

```
# Conduct a KNN classification of the diagnosis variable, set the k parameter to 21(Since 21 is approximately the square root of the training data size). Assign the result to 'diagnosis_knn'.
diagnosis_knn <- knn(train = new_train[-1], test = new_test[-1], cl = new_train$diagnosis, k = 21)

# Calculate the confusion matrix for the test data. Assign the result to the variable 'confusion_knn'.
confusion_knn <- table(test$diagnosis, diagnosis_knn)
confusion_knn
```