Novel Analysis

Licza Lobo

12/12/2024

```r
#Loading in Data from Paper's GitHUB Repository
#(source: https://github.com/kgayvert/PrOCTOR)
PrOCTOR <- readRDS("~/Downloads/PrOCTOR (1).rds")
load("~/Downloads/initial_values.RData")

#Verification of Target-Based Features (Correlation Claims)

View(sample_drugs) #Initial 48 model features for 846 drugs in training set
str(sample_drugs) #data structured as a Matrix

#Converting provided sample drugs into a Data frame
train_set_df <- as.data.frame(sample_drugs)
head(train_set_df)

cor_matrix <- cor(train_set_df)
#This correlation matrix served as an initial analysis and prompted authors to asses further
head(cor_matrix)

# Visualize the correlation matrix
install.packages("corrplot")
library(corrplot)

corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.5)
title(
  main = expression(bold("Figure 1:") ~ "Correlation Matrix Plot of Initial 48 Model Features"),
  cex.main = 1
)
```

```r
#Computing a correlation matrix for the 30 target-based features (median expression values)
#to further confirm the high correlation claim within Target Expression Properties

GTEx_targets_df <- as.data.frame(GTEx_targets)
gtex_cor_matrix <- cor(GTEx_targets_df)

# Visualize the correlation matrix
corrplot(gtex_cor_matrix, method = "color", type = "upper", tl.cex = 0.8)
title(
  main = expression(bold("Figure 2:")
                    ~ "Correlation Matrix Plot of 30 Gene Target Expression Values"),
  cex.main = 1
)
```

```r
#Correlation between 2 sub groups (Chemical and Target-based)

# Split data into chemical and target-based features
chemical_features <- sample_drugs[, 1:10] #all chemical
all_target_features <- sample_drugs[, 14:48] #all target based
exp_target_features <-sample_drugs[,14:43] #only expression target based features


# Compute correlations between chemical and ALL target-based features
cor_matrix_a <- cor(chemical_features, all_target_features)

# View the correlation matrix
print(cor_matrix_a)

# Find the maximum absolute correlation
max_correlation_a <- max(abs(cor_matrix_a))
print(paste("Maximum Pearson's Correlation:", max_correlation_a))


#Compute correlations between chemical and ONLY Expression target-based features
cor_matrix_b <- cor(chemical_features, exp_target_features)

# View the correlation matrix
print(cor_matrix_b)

# Find the maximum absolute correlation
max_correlation_b <- max(abs(cor_matrix_b))
print(paste("Maximum Pearson's Correlation (Just Expression Values:", max_correlation_b))
```

```r
#Completing PCA Analysis to 30 expression values

# Subset columns 14-43 from the sample_drugs data set (only target expression columns)
pca_data <- sample_drugs[, 14:43]

# Check if the data is numeric (required for PCA)
str(pca_data)  # Ensure all columns are numeric

# Scale the selected columns
scaled_data <- scale(pca_data)

# Perform PCA
pca_result <- prcomp(scaled_data, center = TRUE, scale. = TRUE)

# View variance explained
summary(pca_result)

# Scree plot to visualize explained variance
plot(pca_result, type = "l", main = expression(bold("Figure 3:")
              ~ "Scree Plot for PCA of 30 Gene Target Expression Values"),
  cex.main = 1
)

# Extract the first three principal components
```

```r
pca_components <- pca_result$x[, 1:3]
head(pca_components)

# View variance explained by each principal component
summary(pca_result)

# Extract variance proportions for comparison
explained_variance <- pca_result$sdev^2 / sum(pca_result$sdev^2)
cumulative_variance <- cumsum(explained_variance)
print(cumulative_variance[3])


#Correlation between Chemical Properties and PCA1-3 of Gene-target Values
cor_matrix_c <- cor(chemical_features, pca_components)
print(cor_matrix_c)

# Find the maximum absolute correlation
max_correlation_c <- max(abs(cor_matrix_c))
print(paste("Maximum Pearson's Correlation:", max_correlation_c))
```

```r
#Add class labels to sample_drugs data set (provided in paper)

# Create a vector with class labels
drug_class <- c(rep("FTT", 100), rep("FDA Approved", 746))

#New data frame with the original data and class labels
sample_drugs_with_class <- cbind(sample_drugs, drug_class)
write.csv(sample_drugs_with_class, file = "PrOCTOR_training_df.csv") #downloading training set


#Replace expression values with PC1-3 values
pc1_3_values <- pca_result$x[, 1:3]
colnames(pc1_3_values) <- c("PC1", "PC2", "PC3")
sample_drugs_with_class_reduced <- sample_drugs_with_class[, -c(14:43)]
sample_drugs_with_class_final <- cbind(sample_drugs_with_class_reduced, pc1_3_values)


#Train Model on Given Sample Data set

# Separate features and class variable
features <- sample_drugs_with_class_final[, -c(19)]
target <- as.factor(sample_drugs_with_class_final[,c(19)])
```

```r
install.packages("randomForest")

library(randomForest)

#Attempt at Recreating paper's R-Forest approach (30 replicates, 50 bootstrapped subsets)
approved_indices <- which(target == "FDA Approved")
ftt_indices <- which(target == "FTT")
num_replicates <- 30

# Initialize predictions
```

```r
all_predictions <- matrix(0, nrow = length(target), ncol = num_replicates)

# Repeat sub-sampling and training
for (i in 1:num_replicates) {
  sampled_approved <- sample(approved_indices, length(ftt_indices), replace = FALSE)
  sampled_indices <- c(ftt_indices, sampled_approved)

  # Create balanced training set (paper method unclear - will sample 100 for all 100 Failed drugs)
  balanced_features <- features[sampled_indices, ]
  balanced_target <- target[sampled_indices]

  # Train random forest
  rf_model <- randomForest(x = balanced_features, y = balanced_target, ntree = 50)

  # Predict probabilities for the full dataset
  probs <- predict(rf_model, features, type = "prob")

  # Store predictions
  all_predictions[, i] <- probs[, "FDA Approved"]
}


# Calculate average probability for each sample
average_probs <- rowMeans(all_predictions)

# Calculate odds scores
odds_scores <- average_probs / (1 - average_probs)

# Calculate PrOCTOR scores (log2 of odds score)
prOCTOR_scores <- log2(odds_scores)

# Add scores and labels to a data frame for visualization
results_df <- data.frame(
  PrOCTOR_Score = prOCTOR_scores,
  Label = target
)
```

```r
# Plot the distribution of PrOCTOR scores
library(ggplot2)

ggplot(results_df, aes(x = PrOCTOR_Score, fill = Label)) +
  geom_density(alpha = 0.5) +
  labs(
    title = expression(bold("Figure 4:") ~ "Distribution of PrOCTOR Scores"),
    x = "PrOCTOR Score",
    y = "Density"
  ) +
  scale_fill_manual(
    values = c("FDA Approved" = "steelblue", "FTT" = "red") # Customize colors here
  ) +
  theme_minimal()

print(rf_model)
```

```r
#Plot distribution of probability of approval
probability_results_df <- data.frame(
  Average_Probability = average_probs,
  Label = target
)

ggplot(probability_results_df, aes(x = Average_Probability, fill = Label)) +
  geom_density(alpha = 0.5) +
  labs(
    title = expression(bold("Figure 5:") ~ "Distribution of Approval Probabilities"),
    x = "Probability of Approval",
    y = "Density"
  ) +
  scale_fill_manual(
    values = c("FDA Approved" = "steelblue", "FTT" = "red") # Customize colors
  ) +
  theme_minimal()


#KS D Statistics for Seperation of Class (Paper Claims: D=0.5343)
library(dplyr)

# Separate probabilities by class
approved_probs <- probability_results_df %>%
  filter(Label == "FDA Approved") %>%
  pull(Average_Probability)

ftt_probs <- probability_results_df %>%
  filter(Label == "FTT") %>%
  pull(Average_Probability)

# Compute KS statistic
ks_result <- ks.test(approved_probs, ftt_probs)

# Print KS statistic and p-value
print(ks_result)

# Interpretation
cat("Kolmogorov-Smirnov Statistic:", ks_result$statistic, "\n")
cat("P-value:", ks_result$p.value, "\n")
```

```r
#Determine important features and compare to ranking in paper
feature_importance <- importance(rf_model)
print(feature_importance)

#Visualize
importance_df <- data.frame(
  Feature = rownames(feature_importance),
  MeanDecreaseGini = feature_importance[,"MeanDecreaseGini"]
)

# Sort by Mean Decrease Accuracy
importance_df <- importance_df[order(-importance_df$MeanDecreaseGini), ]
```

```r
# Visualize the top features by Mean Decrease Gini
library(ggplot2)

ggplot(importance_df, aes(x = reorder(Feature, -MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(
    title = bold("Figure 6:") ~"Feature Importance (Mean Decrease Gini)",
    x = "Feature",
    y = "Importance (Mean Decrease Gini)"
  ) +
  theme_minimal()
```

```r
#Verifying Kolmogorov-Smirnov (KS) statistic for Features, compare to those in Paper

# Libraries
library(dplyr)

# Separate data into two groups based on class
sample_drugs_with_class_final_df <- as.data.frame(sample_drugs_with_class_final)

sample_approved_df <-
  sample_drugs_with_class_final_df[sample_drugs_with_class_final_df$drug_class == "FDA Approved", ]
sample_ftt_df <-
  sample_drugs_with_class_final_df[sample_drugs_with_class_final_df$drug_class == "FTT", ]

# Initialize an empty data frame to store KS statistics
ks_results <- data.frame(Feature = colnames(features),
                         KS_Statistic = numeric(length(colnames(features))))

# Loop through each feature to compute KS statistic
for (feature in colnames(features)) {
  # Extract feature values for each class
  fda_values <- as.numeric(as.vector(sample_approved_df[[feature]]))
  ftt_values <- as.numeric(as.vector(sample_ftt_df[[feature]]))

  # Compute KS statistic
  ks_stat <- ks.test(fda_values, ftt_values)$statistic
  ks_results[ks_results$Feature == feature, "KS_Statistic"] <- ks_stat
}

print(ks_results)

# Sort features by KS statistic
ks_results_ordered <- ks_results[order(-ks_results$KS_Statistic), ]

# Visualize features by KS statistic
library(ggplot2)
ggplot(ks_results_ordered, aes(x = reorder(Feature, -KS_Statistic), y = KS_Statistic)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = bold("Figure 7:") ~"Features by KS Statistic", x = "Feature", y = "KS Statistic") +
  theme_minimal()
```