Software Tools in Bioinformatics - Assignment 2

Paniz Tayebi

Introduction

Cancer research increasingly relies on bioinformatics tools to explore complex genetic relationships and molecular mechanisms. The cell cycle regulator (CCR) and receptor tyrosine kinase (RTK) pathways are two critical regulators frequently implicated in cancer progression. Understanding gene sequence variations among these pathways can reveal insights into cancer biology, contributing to the development of targeted therapies (Luo et al., 2009). In this project, we analyze gene clusters involved in cell cycle regulation and RTK signaling, focusing on specific genes such as “CCND1”, “RB1”, “CDC25A”, “CCNE1”, “PDGFRA”, “CD117”, “FGFR2”, and “AXL”. By leveraging the vast sequence resources in the National Center for Biotechnology Information (NCBI) database, we aim to establish a machine learning model that distinguishes between these gene groups based on sequence patterns, which could ultimately aid in identifying biomarkers for cancer types linked to these pathways (Hanahan & Weinberg, 2011).

To achieve this, we employ a k-mer frequency approach to represent DNA sequence features numerically, allowing us to apply machine learning techniques. The Random Forest classifier, a widely used supervised learning algorithm, is chosen due to its ability to handle high-dimensional data and provide interpretable feature importance. The RF model works by aggregating the results of multiple decision trees, which enables it to handle high-dimensional data effectively (Breiman, 2001). After pre-processing the sequences to ensure quality and remove biases, the model is trained and evaluated to classify the genes into CCRs and RTKs. This study’s objective is to assess the effectiveness of k-mer-based models in discriminating gene groups with distinct biological roles, potentially advancing bioinformatic methods in cancer research.

Code Section 1

```
## Packages Used ####
library(tidyverse)
library(dplyr)
library(rentrez) # To fetch sequences from NCBI
library(Biostrings) # To manipulate DNA sequences
library(randomForest) # Machine Learning Model
library(ggplot2) # To produce plots and visualization
library(styler) # To tidy code formatting
#library(caret) # Machine learning Model
library(rstudioapi) # provides additional functions for working in RStudio
#library(pROC) # To create the ROC curve and calculate AUC.
```

```
#First I set working directory.
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
# In order to keep formatting consistent I used "styler" package.
styler::style_file[redacted])

# I have created two gene clusters: The first group are Cell Cycle Regulators (CCRs) consisting of CCND1, RB1, CDC25A, and CCNE1 genes. The second group are Receptor Tyrosine Kinases (RTKs) consisting of PDGFRA, CD117, FGFR2, and AXL genes. The objective of this code is to build a classification model using supervised machine learning that distinguishes these two cancer-producing groups of genes based on the obtained sequences.

#From each group, genes with various lengths were selected to ensure that the model captures a realistic range of sequence variability. This also helps to avoid length related biases.

#The k-mer frequency features capture local sequence composition, which can help the model to focus on functionally relevant patterns rather than sequence length.

## 1. Data Acquisition ####

# First I explore the various databases available in NCBI using the "rentrez" package which helped me decide what I should use for this project.

#entrez_dbs() # To get a list of all NCBI databases.
#entrez_db_searchable(db = "nuccore")  #List NCBI databases and searchable fields in 'nuccore' to ensure correct query.

# Then I search for CCR gene sequences (e.g., CCND1, RB1, CDC25A, CCNE1) in mammals within 1500-5000 bp length.
# The retmax for both sequences were determined based on the maximum hits available.

#CCR_Seq <- entrez_search(
#  db = "nuccore",
#  term = "((CCND1[Gene Name] OR RB1[Gene Name] OR CDC25A[Gene Name] OR CCNE1[Gene Name])) AND mammals[Organism] AND 1500:5000 [SLEN]",
#  retmax=2500,
#  use_history = TRUE)

# Then I explore the data:
#class(CCR_Seq)
#CCR_Seq
```

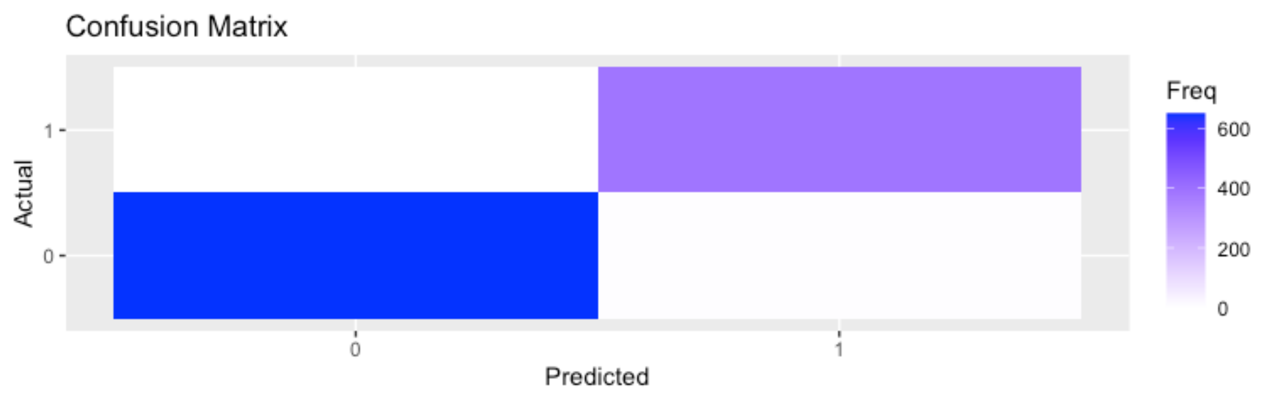
```
#print(CCR_Seq$ids) # Unique IDs
#class(CCR_Seq$ids)
#length(CCR_Seq$ids)
#print(CCR_Seq$count) #Total Hits
#print(CCR_Seq$retmax) #Maximum Hits returned

# Here I repeat the search for RTK genes (e.g., PDGFRA, CD117, FGFR2, AXL) in mammals
within 1500-5000 bp
#RTK_Seq <- entrez_search(
#  db = "nuccore",
#  term = "((PDGFRA[Gene Name] OR CD117[Gene Name] OR FGFR2[Gene Name] OR
AXL[Gene Name])) AND mammals[Organism] AND 1500:5000 [SLEN]",
#  retmax=2500,
#  use_history = TRUE)
#RTK_Seq
#RTK_Seq$ids
#class(RTK_Seq$ids)
#length(RTK_Seq$ids)
#RTK_Seq$count
#RTK_Seq$retmax

# Here I fetch sequences for both searches using history web object, retrieve as FASTA.
Web history is used since the data set was large.
#CCR_fasta <- entrez_fetch(db = "nuccore",
#                      web_history = CCR_Seq$web_history,
#                      rettype = "fasta",
#                      retmode = "text",
#                      count=2500)
#RTK_fasta <- entrez_fetch(db = "nuccore",
#                     web_history =  RTK_Seq$web_history,
#                      rettype = "fasta",
#                     retmode = "text",
#                      count = 2500)

#cat(CCR_fasta)
#cat(RTK_fasta)

# In this part I save the fetched sequences to FASTA files for future access.
#write(CCR_fasta,file = "CCR.fasta")
#write(RTK_fasta,file = "RTK.fasta")
#list.files()

# Here I import sequences into R using "Biostrings" package for DNA sequence analysis.
CCR_sequences <- readDNAStringSet("CCR.fasta", format = "fasta")
```

```
RTK_sequences <- readDNAStringSet("RTK.fasta", format = "fasta")

print(CCR_sequences)
print(RTK_sequences)

# 2.Filtering and Data Control ####

# In this part I clean and filter through the sequences:
# First I remove duplicate sequences and check lengths before and after to see how the
data set was affected.
CCR_uniq <- unique(CCR_sequences)
length(CCR_sequences)
length(CCR_uniq)
RTK_uniq <- unique(RTK_sequences)
length(RTK_sequences)
length(RTK_uniq)

# Then I removed low-quality sequences by removing sequences containing Ns. Here I
checked the lengths again to check the impact on the data.
CCR_clean <- CCR_uniq[!grepl("N", CCR_uniq)]
length(CCR_clean)
RTK_clean <- RTK_uniq[!grepl("N", RTK_uniq)]
length(RTK_clean)

#Then I check sequence lengths and filter by minimum length (e.g., min_length = 350)
#(I did this step and noticed due to prior filtering the data set didn't change so I kept it as it
was.) Filtering or Trimming based on min and max length for this data set is not suitable
since various genes included in each group have non-overlapping length sizes.
#min_length <- 350
#ccr_sequences <- CCR_clean[width(CCR_clean) > min_length]
#rtk_sequences <- RTK_clean[width(RTK_clean) > min_length]
#ccr_sequences
#rtk_sequences
#rm(ccr_sequences)
#rm(min_length)

#Then I plot histogram of sequence lengths for quality control of data.
ggplot(data.frame(Length = width(CCR_clean)), aes(x = Length)) +
  geom_histogram(binwidth = 100, fill = "blue", color = "black") +
  ggtitle("CCR Sequence Length Distribution") +
  xlab("Sequence Length") +
  ylab("Frequency")
ggplot(data.frame(Length = width(RTK_clean)), aes(x = Length)) +
  geom_histogram(binwidth = 100, fill = "blue", color = "black") +
  ggtitle("CCR Sequence Length Distribution") +
  xlab("Sequence Length") +
  ylab("Frequency")

# The output shows various peaks at certain lengths which is mostly consistent with the
reported bp count of each gene from ensembl.org.
```

Code Section 2

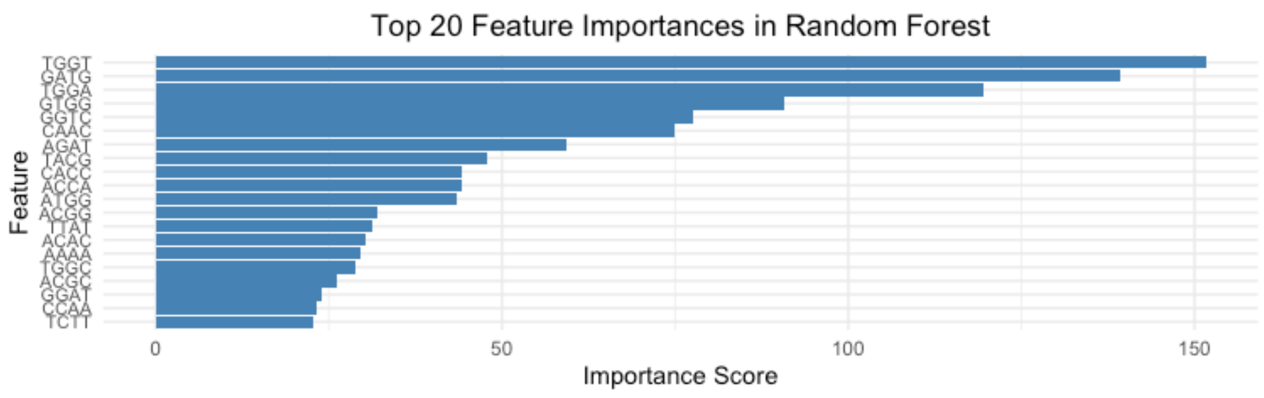
```
# 3.k-mer Frequencies ####
# Now that data exploration, cleaning and modification is done. I am obtaining k-mer
frequencies as a feature for numerical representation of DNA sequences for machine
learning analysis. K-mer frequencies functions as a feature vector for each sequence,
which the model can use as input. This transformation of sequence data into feature
vectors is essential for applying machine learning to genomic sequence classification
tasks.

# Calculating k-mer frequencies (4-mers) for both CCR and RTK sequences
CCR_kmers <- oligonucleotideFrequency(CCR_clean, width=4, step=1)
RTK_kmers <- oligonucleotideFrequency(RTK_clean, width=4, step=1)
CCR_kmers
RTK_kmers

# Combining k-mer frequencies into a data frame for classifier input
kmer_data <- rbind(CCR_kmers, RTK_kmers)
labels <- c(rep(1, nrow(CCR_kmers)), rep(0, nrow(RTK_kmers)))

# 4.Classification Model: Random Forest ####

# First I randomly sample data to split them into training and test sets (80% training, 20%
testing)
set.seed(555)
train_index <- sample(1:nrow(kmer_data), 0.8 * nrow(kmer_data))
train_data <- kmer_data[train_index, ]
train_data
train_labels <- labels[train_index]
test_data <- kmer_data[-train_index, ]
test_labels <- labels[-train_index]

# Using the training sets I then build Random Forest classifier.
model <- randomForest(x = train_data, y = as.factor(train_labels))

# 5.Model Evaluation ####
```

```r
# Now that the classifier model is generated I use the test data to evaluate its percision and accuracy.
# First I generate predictions and calculate accuracy. I also generate a confusion matrix to compare predicted and actual classifications. The table provides a quick preview of the model's performance.
predictions <- predict(model, test_data)
conf_mat <- table(Predicted = predictions, Actual = test_labels)
accuracy <- mean(predictions == test_labels)
print(paste("Accuracy:", accuracy))

# Then I calculate accuracy in a different way, using the confusion matrix and the accuracy formula.
accuracy2 <- sum(diag(conf_mat)) / sum(conf_mat)
cat("Accuracy: ", accuracy, "\n")
conf_mat

#Cross-Evaluation with caret - I tried to perform a secondary evaluation with caret however, the caret_model code wouldn't run. I also attempted to create an ROC curve using "pROC" and "caret”, but I got a bizarre curve.
#rm(train_control)
#rm(predicted_probs)
#rm(roc_curve)

#set.seed(555)
#train_control <- trainControl(method = "cv", number = 10)  # 10-fold cross-validation
#caret_model <- train(
#  x = train_data,
#  y = as.factor(train_labels),
#  method = "rf",  # Random forest model
#  trControl = train_control,
#  importance = TRUE)
#print(caret_model)
#caret_predictions <- predict(caret_model, newdata = test_data)
#caret_conf_mat <- confusionMatrix(caret_predictions, as.factor(test_labels))
#print(caret_conf_mat)

# Generate predicted probabilities for the positive class (assumed to be class 1 here)
#predicted_probs <- predict(model, test_data, type = "prob")[, 2]  # The probability of class 1

# Create the ROC curve
#roc_curve <- roc(test_labels, predicted_probs)

# Plot the ROC curve
# plot(roc_curve, col = "blue", main = "ROC Curve for Random Forest Model",
#    xlim = c(1, 0), ylim = c(0, 1))  # Flip x-axis for specificity
# abline(a = 0, b = 1, lty = 2, col = "gray")  # Diagonal reference line
# cat("AUC:", auc(roc_curve), "\n")

# 6.Visualization ####

# For better visualization I plot confusion matrix as a heat-map using "ggplot2".
ggplot(as.data.frame(conf_mat), aes(x = Predicted, y = Actual, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  ggtitle("Confusion Matrix")
```



```
#The heat-map shows the count of actual vs. predicted classifications. Cells on the diagonal represent correct classifications, while off-diagonal cells represent misclassifications.
#The intensity of the blue color indicates the number of instances in each category.

# 7.Feature Importance Plot ####

# Then in order to create a Feature Importance Plot, I first extract the importance scores from the random forest model in a matrix where each row corresponds to a feature and columns provide different importance metrics and then put them in a data frame with their names and importance scores.

importance <- importance(model)
feature_imp <- data.frame(Feature = rownames(importance), Importance = importance[,1])

# Then I order by importance score and select the top 20 features.
feature_imp <- feature_imp[order(-feature_imp$Importance), ]
top_features <- head(feature_imp, 20)
# Then I generate a bar plot using "ggplot2" to demonstrate the most impactful k-mers.
```

```
ggplot(top_features, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat="identity", fill="steelblue") +
  coord_flip() +
  theme_minimal() +
  ggtitle("Top 20 Feature Importances in Random Forest") +
  xlab("Feature") +
  ylab("Importance Score") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.text.y = element_text(size = 8))
```



```
# 8.Additional Metrics ####

# Here I'm attempting to generate a F1-score since it is a suitable metric used to evaluate the performance of this classification model.

# First I extract counts from the confusion matrix.
TP <- conf_mat[2, 2]  # True Positives
TN <- conf_mat[1, 1]  # True Negatives
FP <- conf_mat[2, 1]  # False Positives
FN <- conf_mat[1, 2]  # False Negatives

# Then I calculate precision, recall, and F1 score.
#Precision or Positive Predictive Value measures the accuracy of positive predictions.
precision <- TP / (TP + FP)
#Recall or Sensitivity or True Positive Rate measures the ability of the model to identify all relevant instances.
recall <- TP / (TP + FN)
#The F1-score combines precision and recall into a single metric.
F1_score <- 2 * (precision * recall) / (precision + recall)
```

```
F1_score
#The F1-score ranges from 0 to 1. 1 indicates perfect precision and recall and 0 indicates the worst performance.

cat("Precision: ", precision, "\n")
cat("Recall: ", recall, "\n")
cat("F1 Score: ", F1_score, "\n")

# Overall the various classifier evaluations show that the generated random forest classifier has a high accuracy score and can classify sequences into the defined labels with precision.
```

Discussion & Conclusion

The findings of this study suggest that k-mer frequency features are effective in capturing sequence-specific information that can distinguish between gene clusters associated with different cancer-related functions. The Random Forest classifier achieved high accuracy in differentiating cell cycle regulators from RTK genes, indicating that specific k-mer patterns are likely associated with the functional differences between these groups. The importance of certain k-mers over others, as reflected in feature importance analysis, may also provide insights into the sequence motifs essential for each gene cluster's role in cancer development and progression.

Despite the promising performance of the model, there are limitations to consider. The sequences used in this study are restricted to certain length ranges, which may not fully capture the diversity of gene variants across different cancer types. Additionally, the absence of cross-evaluation methods such as ROC-AUC analysis in this study, due to technical issues, limits the robustness of the classifier's performance assessment. Future work could benefit from incorporating a larger dataset, exploring additional machine learning models, and including cross-validation techniques to further validate the model's accuracy and generalizability. Overall, this study demonstrates the potential of using k-mer-based models and machine learning in bioinformatics to categorize genes and uncover relationships in cancer-related pathways..

Acknowledgements

I would like to extend my heartfelt gratitude to everyone who contributed to the successful completion of this project. I am especially thankful to Coleman Eagle Olenick for his invaluable assistance in constructing the original database, which laid the groundwork for this assignment. I am deeply appreciative of Yasmine Hezema, whose help with troubleshooting my code was instrumental. I am also grateful to Maryam Allahyari and

Koosha Rokhzadi for their insightful suggestions on improving this assignment. My sincere thanks go to Brittany MacIntyre for her guidance through constructing the step-by-step coding process for this assignment. Finally, I would like to acknowledge ChatGPT, which provided helpful suggestions on achieving specific outcomes and refining my approach. Thank you all for your support and contributions.

References

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/a:1010933404324

Hanahan, D., & Weinberg, Robert A. (2011). Hallmarks of cancer: the next Generation. *Cell*, *144*(5), 646–674. https://doi.org/10.1016/j.cell.2011.02.013

Luo, J., Solimini, N. L., & Elledge, S. J. (2009). Principles of Cancer Therapy: Oncogene and Non-oncogene Addiction. *Cell*, *136*(5), 823–837. https://doi.org/10.1016/j.cell.2009.02.024