به نام خدا

پروژه پایانی درس مبانی انفورماتیک زیستی

تحلیل و بصری‌سازی تغییرات تعداد کپی (CNV) در ژنوم انسان

اعضای گروه:

هلیا شمس‌زاده امیری

بهاره کاوسی نژاد

مهدی معدنی‌پور

بهمن 1403

مستندات پروژه: تغییرات تعداد کپی (CNV) و ارتباط آن با ژن‌ها

بررسی کلی پروژه

این پروژه به تحلیل تغییرات تعداد کپی DNA (CNVs) و ارتباط احتمالی آن‌ها با ژن‌های خاص می‌پردازد. اهداف شامل شناسایی الگوهای CNVs در داده‌های مورد (case) و کنترل، انجام تحلیل‌های آماری برای برجسته‌سازی نواحی مهم CNV، نگاشت این نواحی به ژن‌ها و استفاده از یادگیری ماشین برای پیش‌بینی ارتباطات CNV و شناسایی ژن‌های کلیدی مرتبط با بیماری‌ها می‌باشد.

اهداف پروژه

1. شناسایی نواحی آماری معنادار CNV در داده‌های مورد و کنترل.
2. نگاشت نواحی معنادار به ژن‌های همپوشان بر اساس موقعیت‌های کروموزومی.
3. استفاده از یادگیری ماشین برای پیش‌بینی ارتباطات CNV و اولویت‌بندی ژن‌های مرتبط با بیماری.
4. ارائه بصری‌سازی‌ها و بینش‌های مربوط به الگوهای مشاهده‌شده.

داده‌های مورد نیاز

- **داده‌های case**: شامل CNV ها از افراد مبتلا.
- **داده‌های control**: شامل CNV ها از افراد غیرمبتلا.
- **لیست ژن‌ها:** لیستی از ژن‌ها با موقعیت‌های کروموزومی آن‌ها.
- **ستون‌های کلیدی:**

- Control & Case: Chromosome, Start, End, Type, Patient_ID
- Gene: Gene_ID, Chromosome, Gene_Start, Gene_End

جریان کاری پروژه

پیش‌پردازش داده‌ها

در مرحله اول کتابخانه‌های مورد نیاز (dplyr، tidyr، readr و stats) را بارگذاری (load) می‌کنیم. سپس فایل‌های داده که به فرمت txt هستند را به فرمت csv تبدیل می‌کنیم تا راحت‌تر مورد استفاده قرار گیرند. این فایل‌های csv در پوشه data_CSV قرار گرفته‌اند.

در مرحله بعد ستون‌های داده‌ها را نام‌گذاری و استانداردسازی می‌کنیم. همچنین ستون غیرمرتبط داده را حذف می‌کنیم. این ستون، همان ستون دوم فایل ژن‌ها می‌باشد. اطلاعات این ستون را از ستون سوم نیز می‌توان بدست آورد؛ در نتیجه به آن نیازی نداریم. همچنین یکپارچگی داده‌ها مانند نام‌های کروموزوم و محدوده‌های مختصات را می‌سنجیم.

```
# Load the data
case_data <- read_csv('PC_case.csv', col_names = FALSE)
control_data <- read_csv('PC_control.csv', col_names = FALSE)
gene_list <- read_csv('geneList.csv', col_names = FALSE)

# Rename columns for clarity
colnames(case_data) <- c('Chromosome', 'Start', 'End', 'Type', 'Patient_ID')
colnames(control_data) <- c('Chromosome', 'Start', 'End', 'Type', 'Patient_ID')
colnames(gene_list) <- c('Gene_ID', 'Unused', 'Chromosome', 'Gene_Start',
'Gene_End')
# Drop unused column in gene_list
gene_list <- gene_list %>% select(-Unused)
```

تحلیل آماری

گام 1: تجمیع داده‌ها

در این قسمت داده‌های CNV را بر اساس ستون‌های Chromosome، Start، End و Type گروه‌بندی می‌کنیم تا variationهای یکسان در کنار هم قرار گیرند. همچنین تعداد هرگروه را در case و control می‌شماریم.

```
# Group case and control data by chromosomal regions
case_regions <- case_data %>%
  group_by(Chromosome, Start, End, Type) %>%
  summarise(Case_Count = n(), .groups = 'drop')
```

```
control_regions <- control_data %>%
  group_by(Chromosome, Start, End, Type) %>%
  summarise(Control_Count = n(), .groups = 'drop')
```

گام 2: آزمون دقیق فیشر

آزمون دقیق فیشر یک روش آماری غیرپارامتری است که برای تحلیل جداول فراوانی دو بعدی استفاده می‌شود. این آزمون به‌ویژه زمانی کاربرد دارد که حجم نمونه کوچک باشد و روش‌های تقریبی، مانند آزمون Chi-Square، ممکن است دقت کافی نداشته باشند. آزمون فیشر احتمال مشاهده توزیع داده‌ها در جدول تداخلی را تحت فرض صفر (عدم وجود ارتباط بین دو متغیر) محاسبه می‌کند. این احتمال با استفاده از محاسبات دقیق احتمالاتی انجام می‌شود و برای تعیین معنی‌داری آماری نتایج به کار می‌رود. به دلیل دقت بالای این آزمون، به‌طور گسترده در مطالعات ژنتیکی، زیست‌شناسی و علوم پزشکی برای بررسی ارتباطات بین متغیرها، مانند حضور یا عدم حضور یک ویژگی خاص در گروه‌های مختلف، استفاده می‌شود.

این قسمت از کد برای تحلیل داده‌های تغییرات تعداد کپی (CNV) در بیماران مورد و شاهد طراحی شده است. ابتدا داده‌های مناطق CNV در دو گروه بر اساس کروموزوم، موقعیت شروع، پایان و نوع، ادغام شده و مقادیر خالی به صفر تبدیل می‌شوند. سپس تعداد کل بیماران در هر گروه محاسبه می‌شود. برای هر منطقه، با استفاده از آزمون دقیق فیشر، بررسی می‌شود که آیا حضور تغییرات CNV در گروه مورد به طور معناداری بیشتر از گروه شاهد است یا خیر. نتایج این آزمون به صورت مقدار احتمال (P-Value) ذخیره می‌شوند که نشان‌دهنده معناداری آماری تفاوت‌ها است.

```
# Merge case and control regions
merged_regions <- full_join(case_regions, control_regions, by = c('Chromosome',
'Start', 'End', 'Type')) %>%
  replace_na(list(Case_Count = 0, Control_Count = 0))
# Calculate total case and control patient counts
total_case_patients <- n_distinct(case_data$Patient_ID)
total_control_patients <- n_distinct(control_data$Patient_ID)
# Perform Fisher's exact test
fisher_test <- function(case_count, control_count, total_case, total_control) {
  case_absence <- total_case - case_count
```

```
  control_absence <- total_control - control_count
  contingency_table <- matrix(c(case_count, control_count, case_absence,
control_absence), nrow = 2)
  p_val <- fisher.test(contingency_table, alternative = "greater")$p.value
  return(p_val)
}

merged_regions <- merged_regions %>%
  rowwise() %>%
  mutate(P_Value = fisher_test(Case_Count, Control_Count, total_case_patients,
total_control_patients))
# Filter significant regions
significant_regions <- merged_regions %>%
  filter(P_Value < 0.05)
```

3. نگاشت نواحی معنادار به ژن‌ها

ژن‌های همپوشان برای هر ناحیه CNV معنادار شناسایی شدند و ارتباطات ژنی شناسایی‌شده به‌صورت لیستی که با علامت «;» جدا شده است، ذخیره گردید.

```
# Map significant regions to genes
map_to_genes <- function(chromosome, start, end, genes) {
  overlapping_genes <- genes %>%
    filter(Chromosome == paste0('chr', chromosome),
           Gene_End >= start,
           Gene_Start <= end)
  paste(overlapping_genes$Gene_ID, collapse = ";")
}

significant_regions <- significant_regions %>%
  rowwise() %>%
  mutate(Associated_Genes = map_to_genes(Chromosome, Start, End, gene_list))
# Save results to a CSV file
write_csv(significant_regions, 'Significant_Regions_with_Genes.csv')

# Print completion message
print("Significant regions with associated genes saved to
'Significant_Regions_with_Genes.csv'.")
```

در ابتدا، تابع `map_to_genes` بررسی می‌کند که هر منطقه در کدام کروموزوم قرار دارد و بازه شروع و پایان آن چیست. سپس ژن‌هایی که موقعیت آن‌ها با بازه منطقه همپوشانی دارند، از

دیتافریم ژن‌ها استخراج می‌شوند. شناسه ژن‌های همپوشان به صورت رشته‌ای که با ";" جدا شده است، بازگردانده می‌شود. این تابع برای تمام مناطق مهم اعمال شده و ژن‌های مرتبط در ستون جدیدی به نام `Associated_Genes` ذخیره می‌شوند. در نهایت، داده‌های مناطق مهم به همراه ژن‌های مرتبط در یک فایل CSV ذخیره می‌شوند .

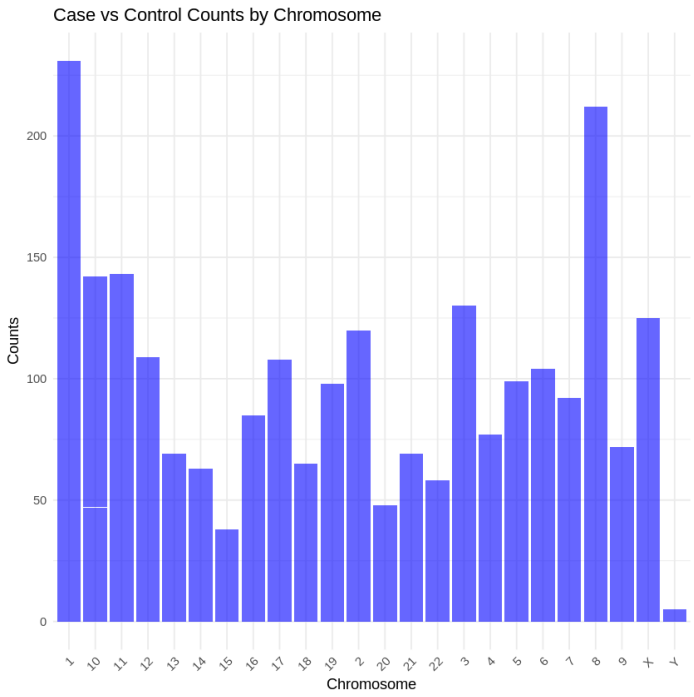
5. بصری‌سازی نتایج

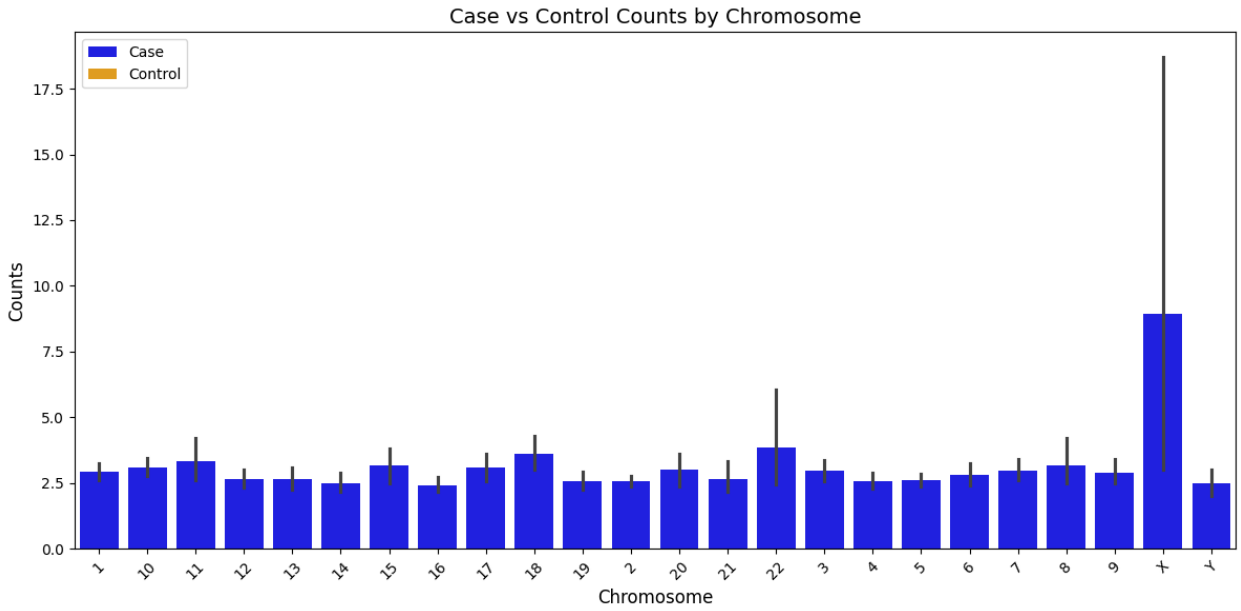
```
# Visualizations
library(ggplot2)
```

نمودار میله‌ای: تعداد موارد در مقابل کنترل‌ها بر اساس کروموزوم

```
# Bar Plot: Case vs Control Counts
ggplot(significant_regions, aes(x = Chromosome)) +
  geom_bar(aes(y = Case_Count), stat = "identity", fill = "blue", alpha = 0.6) +
  geom_bar(aes(y = Control_Count), stat = "identity", fill = "orange", alpha =
0.6) +
  labs(title = "Case vs Control Counts by Chromosome", y = "Counts", x =
"Chromosome") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

نتایج:



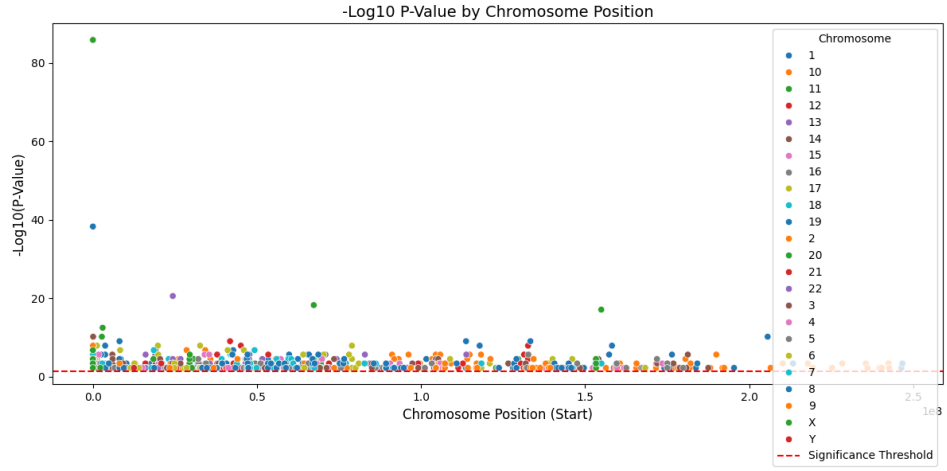


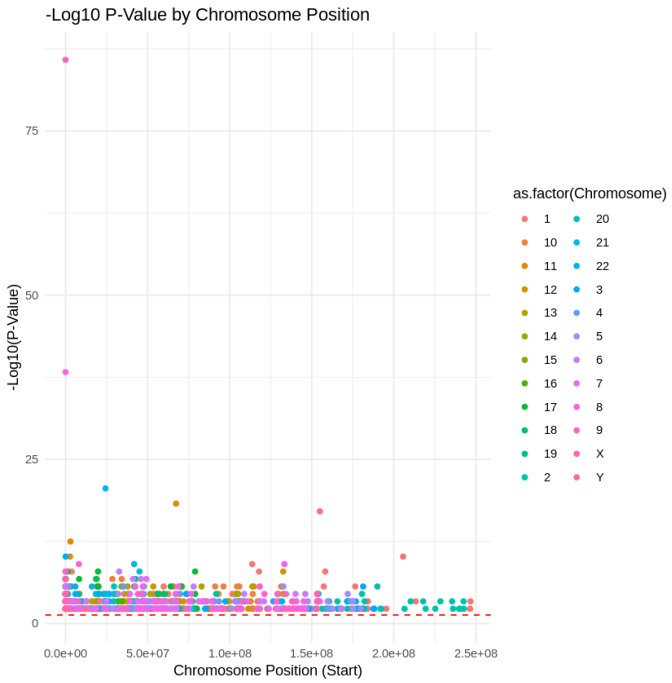
نمودار پراکندگی: موقعیت کروموزومی در مقابل مقدار P

```
# Scatter Plot: Chromosome Position vs P-Value
significant_regions <- significant_regions %>%
  mutate(Log_P_Value = -log10(P_Value))

ggplot(significant_regions, aes(x = Start, y = Log_P_Value, color =
as.factor(Chromosome))) +
  geom_point() +
  geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "red") +
  labs(title = "-Log10 P-Value by Chromosome Position", y = "-Log10(P-Value)", x
= "Chromosome Position (Start)") +
  theme_minimal()
```

نتایج:





استفاده از یادگیری ماشین

ایجاد ویژگی‌ها

برای تحلیل داده‌ها، ابتدا داده‌های مربوط به case و control ترکیب شدند، به‌طوری که یک ستون هدف (target) برای مشخص‌کردن caseها (با مقدار 1) و کنترل‌ها (با مقدار 0) به هر مجموعه داده اضافه شد. سپس، این دو مجموعه داده با استفاده از تابع bind_rows با یکدیگر ادغام شدند. در ادامه، برای هر رکورد در داده‌های ترکیبی، ژن‌های همپوشان شناسایی شدند. این کار با تطبیق کروموزوم، موقعیت شروع و پایان هر ناحیه با اطلاعات ژنوم در داده‌های ژنی انجام شد. ژن‌های شناسایی‌شده برای هر رکورد به‌صورت لیستی که با علامت «;» از یکدیگر جدا شده‌اند، در ستون جدیدی به نام Associated_Genes ذخیره شدند. در پایان، داده‌های ترکیبی شامل اطلاعات مناطق کروموزومی و ژن‌های مرتبط به‌صورت یک فایل CSV ذخیره شدند. این فایل در پوشه Machine Learning موجود است.

```
# Combine case and control data
case_data <- mutate(case_data, target = 1)
control_data <- mutate(control_data, target = 0)
```

```
combined_data <- bind_rows(case_data, control_data)
# Map significant regions to genes
map_to_genes <- function(chromosome, start, end, genes) {
  overlapping_genes <- genes %>%
    filter(Chromosome == paste0("chr", chromosome),
           Gene_End >= start,
           Gene_Start <= end)
  paste(overlapping_genes$Gene_ID, collapse = ";")
}

combined_data <- combined_data %>%
  rowwise() %>%
  mutate(Associated_Genes = map_to_genes(Chromosome, Start, End,
gene_data))
# Export the combined data to a CSV file
write_csv(combined_data, "combined_data.csv")
```

برای بهبود تحلیل داده‌ها، ابتدا مهندسی ویژگی انجام شد که در آن یک ویژگی جدید به نام VariationLength محاسبه شد که اختلاف بین موقعیت پایان (End) و موقعیت شروع (Start) را نشان می‌دهد. این ویژگی به‌عنوان طول ناحیه کروموزومی متغیر (CNV) به داده‌ها اضافه شد. همچنین، ستون مربوط به شناسه بیماران (Patient_ID) حذف شد، زیرا برای تحلیل موردنیاز نبود. در مرحله بعد، برای آماده‌سازی داده‌ها جهت مدل‌سازی، ستون‌های اسمی شامل Chromosome، Type و Associated_Genes به مقادیر فاکتوری تبدیل شدند و سپس از تکنیک One-Hot Encoding برای تبدیل مقادیر دسته‌بندی‌شده به متغیرهای عددی دودویی استفاده شد. این تبدیل با استفاده از تابع model.matrix انجام شد و داده‌های پردازش‌شده به‌صورت یک فریم داده قابل‌استفاده برای مدل‌سازی ذخیره شدند.

```
# Feature engineering
combined_data <- combined_data %>%
  mutate(VariationLength = End - Start) %>%
  select(-Patient_ID)
# One-Hot Encoding
encoded_data <- combined_data %>%
  mutate(across(c(Chromosome, Type, Associated_Genes), as.factor)) %>%
  model.matrix(~ . - 1, data = .) %>%
  as.data.frame()
```

آموزش مدل

در این مرحله، مدل یادگیری ماشین برای پیش‌بینی ایجاد شد. ابتدا با استفاده از داده‌های آماده‌شده، متغیرهای مستقل (X) و متغیر هدف (y) از داده‌ها جدا شدند. متغیر y شامل مقادیر 1 برای caseها و 0 برای کنترل‌ها (control) بود. سپس با استفاده از تابع createDataPartition از کتابخانه caret، داده‌ها به دو مجموعه آموزشی و آزمایشی تقسیم شدند، به‌طوری که 80 درصد داده‌ها به‌عنوان مجموعه آموزشی (X_train و y_train) و 20 درصد باقی‌مانده به‌عنوان مجموعه آزمایشی (X_test و y_test) انتخاب شدند.

برای مدل‌سازی، از الگوریتم جنگل تصادفی (Random Forest) استفاده شد. این الگوریتم با استفاده از مجموعه آموزشی (X_train و y_train) آموزش داده شد و هدف آن یادگیری الگوهای موجود در داده‌ها برای پیش‌بینی مقادیر هدف بود. از set.seed(42) نیز برای اطمینان از تکرارپذیری نتایج استفاده شد. این مرحله پایه‌ای برای ارزیابی و استفاده از مدل در مراحل بعدی فراهم می‌کند.

```
# Train a model
set.seed(42)
X <- encoded_data %>% select(-target)
y <- encoded_data$target
train_index <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[train_index, ]
X_test <- X[-train_index, ]
y_train <- y[train_index]
y_test <- y[-train_index]

model <- randomForest(X_train, as.factor(y_train))
```

برای ارزیابی عملکرد مدل جنگل تصادفی که در مرحله قبلی آموزش داده شد، پیش‌بینی‌هایی روی مجموعه آزمایشی (X_test) انجام شد و نتایج پیش‌بینی‌شده (y_pred) با مقادیر واقعی (y_test) مقایسه شدند. این مقایسه با استفاده از Confusion Matrix انجام شد که معیارهایی مانند تعداد پیش‌بینی‌های صحیح و غلط برای هر کلاس (case و کنترل‌ها) را نشان می‌دهد. ماتریس سردرگمی اطلاعات جامعی از جمله دقت (Accuracy)، حساسیت (Sensitivity)، ویژگی (Specificity) و دیگر معیارهای مرتبط را ارائه می‌دهد که برای ارزیابی دقت و قابلیت مدل در تمایز بین کلاس‌ها

بسیار مفید است. این مرحله ارزیابی، به درک بهتر عملکرد مدل روی داده‌هایی که قبلاً در آموزش استفاده نشده‌اند کمک می‌کند.

```
# Evaluate the model
y_pred <- predict(model, X_test)
confusion_matrix <- confusionMatrix(as.factor(y_pred), as.factor(y_test))
print(confusion_matrix)
```

در این مرحله، Feature Importance برای مدل جنگل تصادفی محاسبه شد. این موضوع نشان می‌دهد که هر ویژگی تا چه اندازه در پیش‌بینی‌های مدل تأثیرگذار بوده است. با استفاده از تابع importance، میزان تأثیر هر ویژگی بر تصمیم‌گیری مدل محاسبه شد و سپس این مقادیر در یک فریم داده جدید به نام feature_importance ذخیره شدند. این فریم داده شامل نام ویژگی‌ها (Feature) و مقادیر اهمیت آنها (Importance) است. برای شفافیت بیشتر، ویژگی‌ها بر اساس اهمیت آنها به ترتیب نزولی مرتب شدند. این اطلاعات به شناسایی ویژگی‌های کلیدی کمک می‌کند و می‌تواند در بهبود مدل یا تفسیر نتایج مورد استفاده قرار گیرد.

```
# Feature importance
feature_importance <- data.frame(Feature = colnames(X_train), Importance =
importance(model)) %>%
  arrange(desc(Importance))
```

در این بخش، ژن‌های مهم مرتبط با case شناسایی شدند. ابتدا ستون‌هایی که شامل اطلاعات مربوط به ژن‌های مرتبط بودند (ستون‌هایی که با _Associated_Genes شروع می‌شدند)، شناسایی شدند. سپس، داده‌های مربوط به caseها (target == 1) فیلتر شده و مجموع مقادیر هر ژن در این ستون‌ها محاسبه شد. نتایج به‌صورت یک فریم داده با نام ژن‌ها (Gene) و تعداد رخدادهای آنها (Count) ذخیره و بر اساس تعداد به ترتیب نزولی مرتب شدند. این اطلاعات در فایل significant_genes.csv ذخیره شد.

برای تجزیه و تحلیل بیشتر، ژن‌ها در ستون مربوطه بر اساس «;» جدا شدند و شمارش مجددی برای هر ژن انجام شد تا تعداد کل رخدادهای هر ژن در داده‌ها مشخص شود. این اطلاعات نیز در فایل gene_counts.csv ذخیره شدند.

در نهایت، برای ارائه بصری، ۲۰ ژن برتر با بیشترین تعداد رخداد به‌صورت نمودار میله‌ای نمایش داده شدند. این نمودار، تعداد رخدادهای هر ژن را در محور عمودی و نام ژن‌ها را در محور افقی نشان می‌دهد، و ترتیب ژن‌ها به‌گونه‌ای تنظیم شده که ژن‌هایی با بیشترین تعداد رخداد در بالای نمودار قرار گیرند. این مرحله به شناسایی ژن‌های کلیدی در ارتباط با موارد کمک می‌کند.

```
# Identify significant genes
gene_columns <- grep("Associated_Genes_", colnames(encoded_data), value =
TRUE)
significant_genes <- encoded_data %>%
  filter(target == 1) %>%
  select(all_of(gene_columns)) %>%
  summarise(across(everything(), sum)) %>%
  pivot_longer(everything(), names_to = "Gene", values_to = "Count") %>%
  arrange(desc(Count))

write_csv(significant_genes, "significant_genes.csv")
# Gene counts
significant_genes <- read_csv("significant_genes.csv")
# Split the genes and count occurrences
gene_counts <- significant_genes %>%
  separate_rows(Gene, sep = ";") %>%
  count(Gene, name = "Count") %>%
  arrange(desc(Count))

write_csv(gene_counts, "gene_counts.csv")
# Plot the top 20 genes with the highest counts
top_genes <- head(gene_counts, 20)

ggplot(top_genes, aes(x = reorder(Gene, Count), y = Count)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "Top 20 Genes with Highest Counts", x = "Gene", y =
"Count") +
  theme_minimal()
```

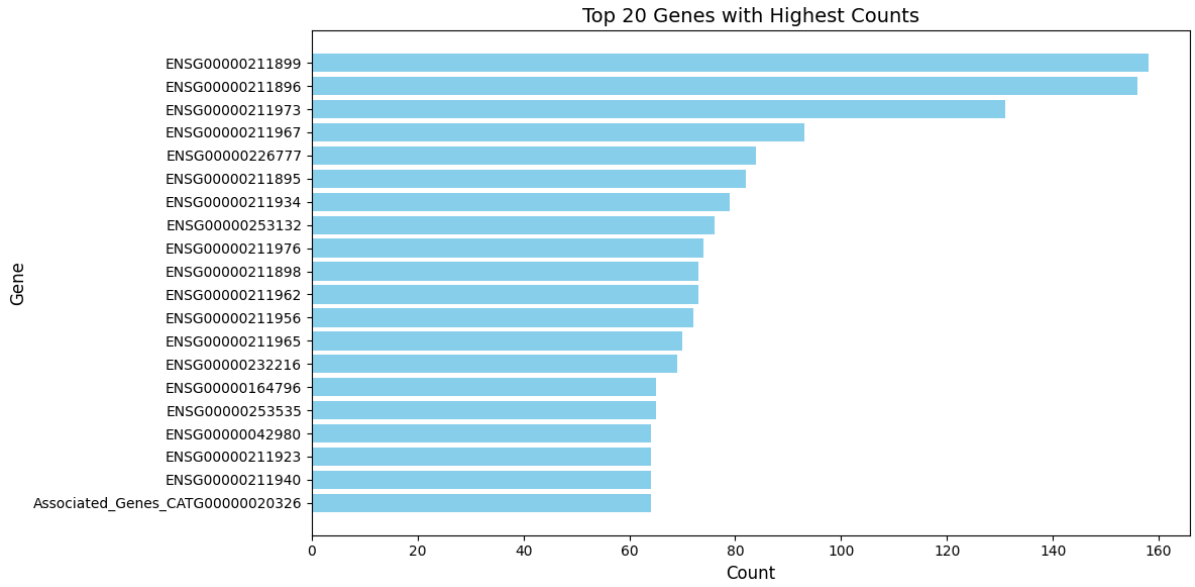
به طور خلاصه، برای ایجاد و ارزیابی یک مدل یادگیری ماشین با استفاده از Random Forest Classifierابتدا داده‌های ورودی (ویژگی‌ها) و خروجی (هدف) از دیتافریم combined_data_encoded استخراج می‌شوند. ستون هدف (target) به عنوان برچسب

کلاس مشخص شده و سایر ستون‌ها به عنوان ویژگی‌ها (X) در نظر گرفته می‌شوند. سپس داده‌ها به دو بخش آموزش (80%) و آزمون (20%) تقسیم می‌شوند.

مدل Random Forest Classifierایجاد شده و با داده‌های آموزشی ، آموزش داده می‌شود. پس از آن، مدل برای پیش‌بینی کلاس‌ها در داده‌های آزمون (X_test) استفاده می‌شود و خروجی پیش‌بینی‌شده (y_pred) با مقادیر واقعی (y_test) مقایسه می‌شود. در نهایت، متریک‌های ارزیابی مدل شامل دقت(Precision) ، بازخوانی(Recall) ، امتیاز F1 و دقت کل (Accuracy) با استفاده از تابع classification_reportچاپ می‌شوند. این ارزیابی نشان می‌دهد که مدل چقدر در دسته‌بندی داده‌ها عملکرد خوبی داشته است.

نتایج:

	precision	recall	f1-score	support
0	0.98	0.99	0.99	6475
1	0.97	0.96	0.97	2328
accuracy			0.98	8803
macro avg	0.98	0.97	0.98	8803
weighted avg	0.98	0.98	0.98	8803



بر اساس این نمودار متوجه می‌شویم که ژن ENSG00000211899 بیشترین تکرار را داشته و احتمالاً در بروز بیماری موثر است.