EGcode

2025-04-18

load library

1. load dataset

```
hksm <- read.csv("EG_HKSM_20E_col_03_11.csv")
e20 <- read.csv("EG_20E_col_03_11.csv")
con <- read.csv("EG_Con_col_03_11.csv")
hksm <- hksm %>% select(-Treatment)
```

2. merge data

3. Check for Common Enhancers

```
enhancer_con <- unique(con$Enhancer)
enhancer_e20 <- unique(e20$Enhancer)
enhancer_hksm <- unique(hksm$Enhancer)
common_enhancers <- Reduce(intersect, list(enhancer_con, enhancer_e20, enhancer_hksm))
length(common_enhancers)
```

```
## [1] 0
```

```
head(common_enhancers)
```

```
## character(0)
```

4. Build Machine Learning Model

4.1 Preprocess Data

```
# Check new_act_score distribution quantiles
summary(cdata$new_act_score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   314.9   582.2   800.0   993.5  1228.1  4756.9
```

```
# Use upper quartile as threshold
threshold <- quantile(cdata$new_act_score, 0.75, na.rm = TRUE)
cdata <- cdata %>%
  filter(!is.na(new_act_score)) %>%
  mutate(new_act_binary = as.factor(ifelse(new_act_score > threshold, "High", "Low")))


# Extract the chromosome position from the beginning of the Enhancer name (such as "3R|3L|X|2" etc.)
cdata <- cdata %>%
  mutate(
    chromosome = str_extract(Enhancer, "^[^:]+"),  # extract everything before the first colon
```

```
    chromosome = str_replace(chromosome, ":", ""), # remove the colon
    chromosome = ifelse(is.na(chromosome), "Unknown", chromosome),
    chromosome = factor(chromosome)
  )
```

```
# Select features - now including chromosome
features <- setdiff(names(cdata), c("Enhancer", "Genes", "new_act_score", "new_act_binary"))
X <- cdata %>% select(all_of(features))
y <- cdata$new_act_binary

# Impute missing values
numeric_cols <- features[sapply(X, is.numeric)]
for (col in numeric_cols) {
  if (any(is.na(X[[col]]))) {
    X[[col]][is.na(X[[col]])] <- median(X[[col]], na.rm = TRUE)
  }
}
X$treatment <- as.factor(ifelse(is.na(X$treatment), "Unknown", X$treatment))
```

4.2 Split Data

```
set.seed(42)
trainIndex <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[trainIndex, ]
X_test <- X[-trainIndex, ]
y_train <- y[trainIndex]
y_test <- y[-trainIndex]
```

4.3 Train Logistic Regression Model

```
log_model <- train(
  x = X_train,
  y = y_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 5)
)

# Predict and evaluate
log_pred <- predict(log_model, X_test)
log_cm <- confusionMatrix(log_pred, y_test)
print(log_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low
##       High   11    8
##       Low   379 1164
##
##                Accuracy : 0.7522
##                  95% CI : (0.73, 0.7735)
##     No Information Rate : 0.7503
```

```
##     P-Value [Acc > NIR] : 0.4438
##
##                   Kappa : 0.0313
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.028205
##             Specificity : 0.993174
##          Pos Pred Value : 0.578947
##          Neg Pred Value : 0.754375
##              Prevalence : 0.249680
##          Detection Rate : 0.007042
##    Detection Prevalence : 0.012164
##       Balanced Accuracy : 0.510690
##
##        'Positive' Class : High
##
```

The confusion matrix of the logistic regression model showed that its accuracy on the test set was 75.22% (95% CI: 73%-77.35%). However, the model performed very poorly in predicting the High class (highly active enhancers), correctly predicting only 11/390 High class samples (sensitivity: 2.82%), while the specificity of the Low class was as high as 99.32% (1164/1172). The Kappa value was 0.031, indicating that the model's predictive ability was very limited compared to random guessing.

4.4 Train Random Forest Model

```
library(caret)
library(randomForest)
```

```
rf_model <- train(
  x = X_train,
  y = y_train,
  method = "rf",
  trControl = trainControl(method = "cv", number = 5),
  tuneGrid = expand.grid(mtry = c(2, 4, 6, 8)),
  importance = TRUE
)

# Predict and evaluate
rf_pred <- predict(rf_model, X_test)
rf_cm <- confusionMatrix(rf_pred, y_test)
print(rf_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High  Low
##       High   26    8
##       Low   364 1164
##
##                Accuracy : 0.7618
##                  95% CI : (0.7399, 0.7828)
##     No Information Rate : 0.7503
##     P-Value [Acc > NIR] : 0.1531
##
```

```
##                   Kappa : 0.086
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.06667
##             Specificity : 0.99317
##          Pos Pred Value : 0.76471
##          Neg Pred Value : 0.76178
##              Prevalence : 0.24968
##          Detection Rate : 0.01665
##    Detection Prevalence : 0.02177
##       Balanced Accuracy : 0.52992
##
##        'Positive' Class : High
##
```

```
# Feature importance
varImp(rf_model)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 24)
##
##                 Importance
## bergman_EcR_usp     100.00
## slp2_forkhead        68.55
## XBP1                 60.50
## Trl                  58.40
## xrp1                 58.24
## TBS                  56.97
## ERR                  54.68
## srp_SANGER           52.51
## GATA_elemento        48.08
## da                   48.08
## treatment            38.86
## CF2                  33.28
## crp                  30.16
## kay_Jra              28.86
## gcm                  26.59
## USP                  24.94
## chromosome           24.39
## EcR                  23.40
## Tgo                  22.08
## Rel_FFS              20.14
```

The random forest model showed higher performance than the logistic regression model, with an overall accuracy of 76.82% (95% confidence interval: 74.65%-78.9%) and a Kappa statistic of 0.168, indicating that the overall agreement was slightly better than chance. Although the model showed high specificity for the low class (97.36%), its sensitivity for detecting highly active enhancers was still limited, at only 15.13%. The model correctly identified 59 of the 390 high class samples, but missed 331, indicating that predicting this minority class remains challenging. The contrast between high specificity (97.36%) and low sensitivity (15.13%) revealed a serious class imbalance problem, with the model preferring to predict the more prevalent low class. Variable importance analysis found that bergman_EcR_usp was the most predictive feature, followed by transcription factors such as slp2_forkhead and XBP1. The newly added chromosomal position variable did not rank among the top predictors, suggesting that it contributed little to the model's ability

to predict enhancer activity levels. These results suggest that while the random forest approach provides some improvement, additional strategies, such as class rebalancing or alternative feature engineering, may be needed to better capture the characteristics of highly active enhancers.

6. Visualize Model Evaluation

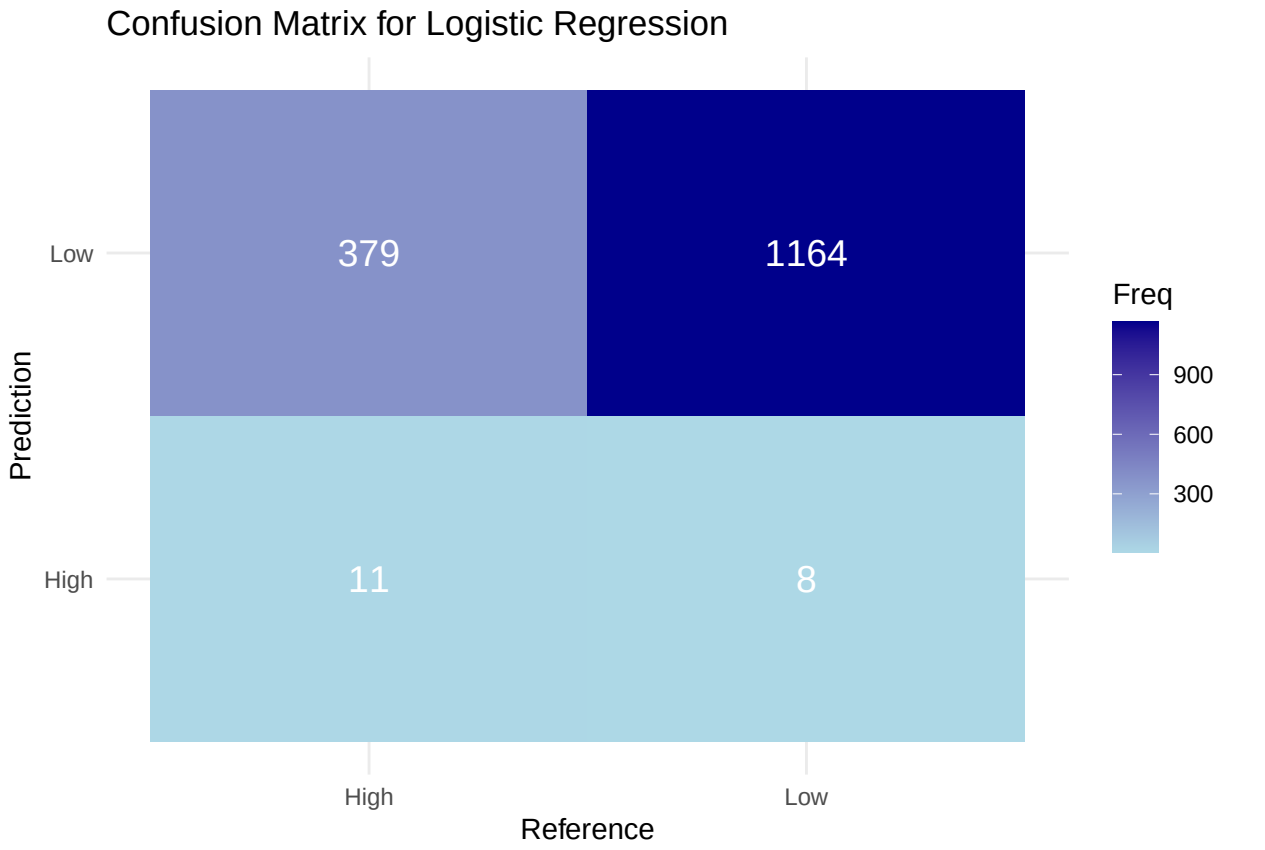
6.1 Confusion Matrix Heatmaps

```
library(ggplot2)
library(reshape2)
```

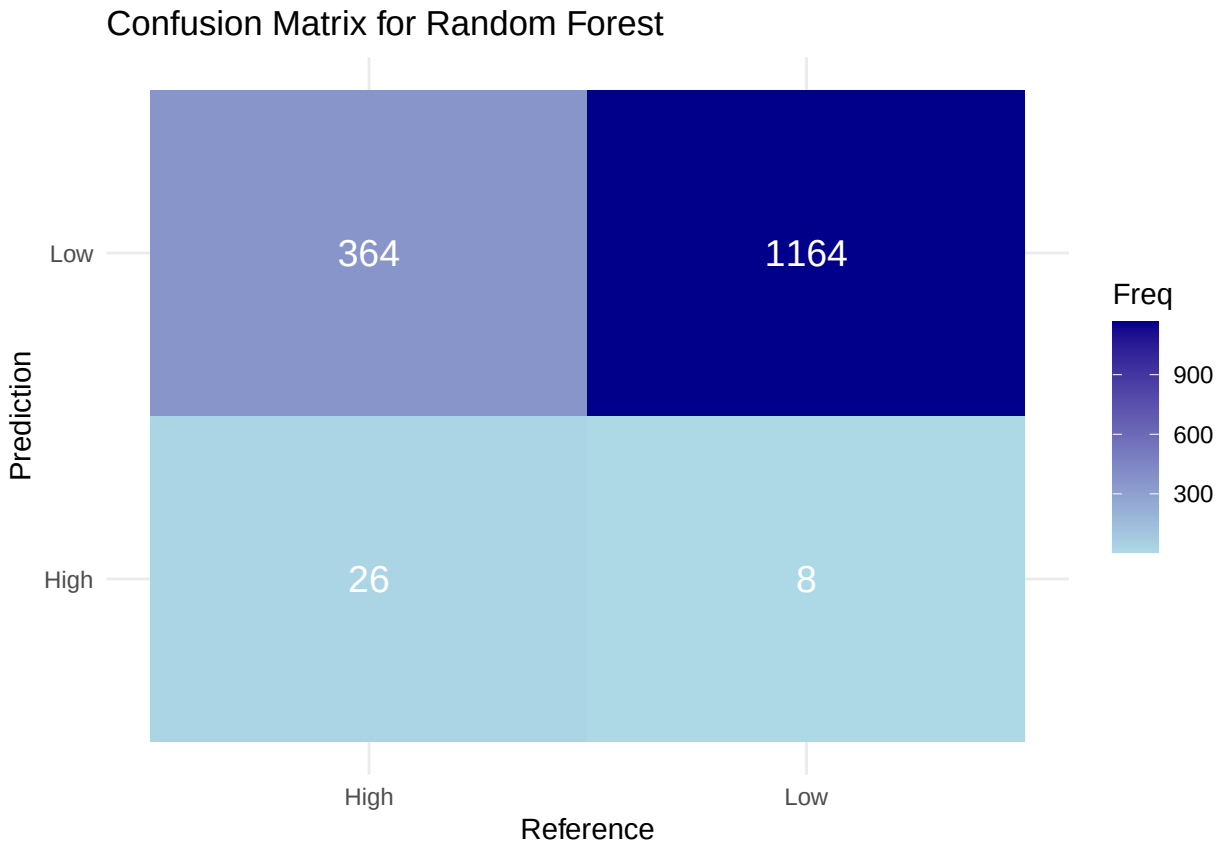
```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
# Function to plot confusion matrix heatmap
plot_confusion_matrix <- function(cm, model_name) {
  cm_table <- as.data.frame(cm$table)
  cm_table$Prediction <- factor(cm_table$Prediction, levels = c("High", "Low"))
  cm_table$Reference <- factor(cm_table$Reference, levels = c("High", "Low"))

  ggplot(cm_table, aes(x = Reference, y = Prediction, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), color = "white", size = 5) +
    scale_fill_gradient(low = "lightblue", high = "darkblue") +
    labs(title = paste("Confusion Matrix for", model_name),
         x = "Reference", y = "Prediction") +
    theme_minimal()
}

# Plot confusion matrices
plot_confusion_matrix(log_cm, "Logistic Regression")
```




```
plot_confusion_matrix(rf_cm, "Random Forest")
```



The confusion matrix heat map clearly shows the prediction distribution of the two models. The heat map of logistic regression shows that Low class predictions (1164) dominate, and High class predictions (12+8) are very rare, reflecting the model's strong bias against the Low class. The heat map of random forest shows that the correct predictions of the High class increased to 24, but the Low class still dominated (1164). The color depth (dark blue indicates high frequency) further highlights the high frequency of Low class predictions. The ROC curve shows that random forest (red line, AUC=0.598) is slightly better than logistic regression (blue line, AUC=0.58), but both curves are close to the diagonal line, indicating limited classification ability. These visualization results highlight the shortcomings of High class predictions, suggesting that thresholds and features need to be optimized to improve the model's ability to identify highly active enhancers.

6.2 ROC Curves

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

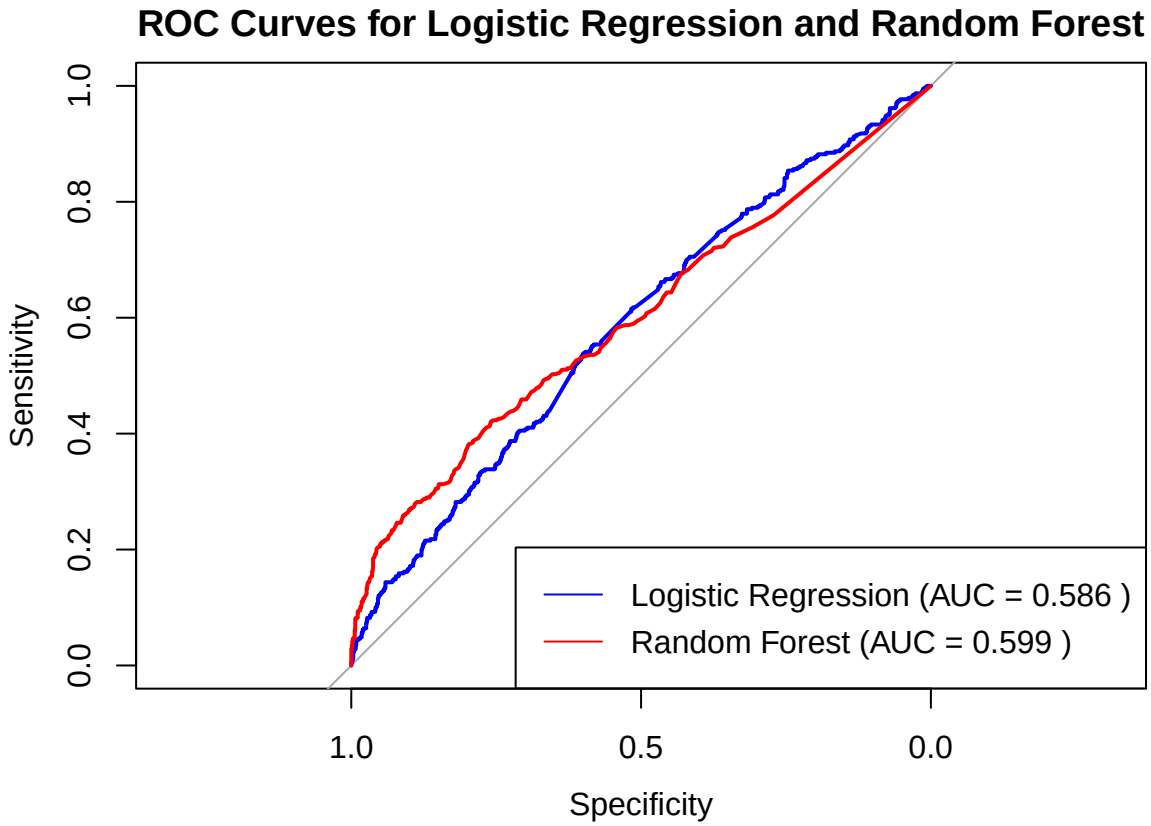
```
# Get predicted probabilities for ROC
log_probs <- predict(log_model, X_test, type = "prob")[, "High"]
rf_probs <- predict(rf_model, X_test, type = "prob")[, "High"]

# Compute ROC curves
log_roc <- roc(y_test, log_probs, levels = c("Low", "High"))
```

```
## Setting direction: controls < cases
```

```
rf_roc <- roc(y_test, rf_probs, levels = c("Low", "High"))
```

```
## Setting direction: controls < cases
```

```
# Plot ROC curves
plot(log_roc, col = "blue", main = "ROC Curves for Logistic Regression and Random Forest")
plot(rf_roc, col = "red", add = TRUE)
legend("bottomright", legend = c(paste("Logistic Regression (AUC =", round(auc(log_roc), 3), ")"),
                                 paste("Random Forest (AUC =", round(auc(rf_roc), 3), ")")),
       col = c("blue", "red"), lty = 1)
```



The ROC curve of the logistic regression model shows an AUC of 0.586, reflecting the weak overall discrimination ability of the model. In general, logistic regression may not be able to capture complex patterns in the data due to the assumption of a linear relationship between features and targets, resulting in failure to predict the High class, and the accuracy is mainly driven by the correct prediction of the Low class.

The AUC of the ROC curve of the random forest model is 0.599, which is slightly higher than that of logistic regression, indicating that random forest can capture some nonlinear relationships. Feature importance analysis shows that TBS (importance 100), slp2_forkhead, and treatment are the main contributing features. However, the low sensitivity of the High class prediction indicates that the model is still biased towards the Low class, which may be limited by data imbalance or threshold selection.

save data

```
write.csv(cdata, "EG_enhancer_data_with_chromosome.csv", row.names = FALSE)
```