Report_Final_Project

Ricardo Arturo Figueroa -Juan Diego Fonseca

2023-11-22

Step 1: Installing the DynamicCancerDriverKM Package Las librerías utilizadas para concluir este proyecto son las siguientes (usar el comando **`library()`**) > tidyverse-caret-class-gmodels-DynamicCancerDriverKM-rpart-randomForest-e1071-kernlab

Nota: La instalación de la librería "DynamicCancerDriverKM" y más información en el siguiente enlace: https://github.com/AndresMCB/DynamicCancerDriverKM

Este **proyecto aplica** los modelos de aprendizaje automático supervisado a un conjunto de datos de expresiones genéticas del cáncer, con el fin de clasificar con precisión distintos tipos de cáncer y predecir resultados clínicos. Consiste en entrenar modelos sobre el conjunto de datos de la librería mencionada anteriormente etiquetando los niveles de expresión de los genes como características y el tipo o estadio de cáncer como objetivo.

Tipo o estadio del cáncer como variable objetivo.

Modelos de aprendizaje: K-nn, Regresión lineal, arboles de decisión, arboles aleatorios, soporte vectorial

Nota: Para visualizar el código, como fases entrenamiento, filtros, selección de genes y construcción de los modelos ver el archivo - `Test.R`

Step 2: Creating a Unified Gene Expression Matrix Se creo una única matriz fusionando los datos de TCGA_BRCA_Normal.rdata y TCGA_BRCA_PT.rdata, esta cuanta con la columna **`Sample_type`** nuestra variable objetivo. De la siguiente manera:

```
normaldata <-(DynamicCancerDriverKM::BRCA_normal)
dataPt <-(DynamicCancerDriverKM::BRCA_PT)
final_data <- bind_rows(normaldata, dataPt)

rate_min_10 <- final_data %>%
               summarise_all(~ mean(. <400, na.rm = TRUE))

colmn_delate <- names(rate_min_10[, rate_min_10 >= 0.8])
data_filter <- final_data %>%
               select(-one_of(colmn_delate))
```

Relacion con los datos de cáncer de mama (BRCA). Se concectan verticalmente los conjuntos de datos usando la función **`bind_rows`** y se almacenan en **`final_data`**. Luego, se calcula la media de cada columna que contiene valores inferiores a 400, y se filtran las columnas que tienen una media superior al 80% de las observaciones.

Realizando una limpieza y filtrado de datos, eliminando las columnas que tienen una alta proporción de valores superiores a 400.

Step 3: Filtering Gene Expression Data La manipulación de datos relacionados con una red de (PPI). Inicialmente, se crea una copia del conjunto de datos original llamado **data_filter** para preservar la integridad del dataset original. Luego, se carga un conjunto de datos PPI desde el paquete **DynamicCancerDriverKM** y se realiza una serie de transformaciones utilizando las funciones de la librería Tidyverse. Se convierten las columnas que representan genes de nodos de entrada y salida en un formato más largo y se calcula la frecuencia de cada combinación. Posteriormente, se organiza la información en un formato más ancho. Además, se crea una nueva variable llamada **total_mode** que representa la suma de las frecuencias de nodos de entrada y salida, y se ordena el conjunto de datos en orden descendente según esta variable.

Se visualiza los primeros 10 datos

```
print(data_PPInR %>% head(10))
```

```
## # A tibble: 10 x 2
## # Groups:   gen [10]
##    gen    total_mode
##    <chr>       <int>
##  1 TP53          299
##  2 CREBBP        273
##  3 EP300         270
##  4 YWHAG         252
##  5 SMAD3         225
##  6 GRB2          210
##  7 SRC           195
##  8 AR            179
##  9 ESR1          174
## 10 RB1           169
```

Step 4: Selecting Genes Based on the PPI Network En este paso se realiza un proceso de manipulación de datos para facilitar la posterior selección y la utilización de un conjunto de predictores en un análisis. Primero, se extraen los nombres de las columnas relevantes del conjunto de datos **data_filter** desde la columna 8 hasta la última y se almacenan en **data_filter_x**. Luego, se utiliza la función **changeGeneId** del paquete **AMCBGeneUtils** para convertir los identificadores de genes en **data_filter_x** desde el formato "Ensembl.ID". Los correspondientes identificadores de símbolos de genes "HGNC.symbol". Los nuevos nombres de genes obtenidos se asignan a las columnas correspondientes en el conjunto de datos original **data_filter**. Después, se extraen los nombres actualizados de las columnas y se almacenan en **final_data_gen**.

A continuación, se filtran las filas del conjunto de datos **data_PPInR** para incluir solo aquellas cuyos nombres de genes coinciden con los nombres actualizados presentes en **final_data_gen**, resultando en un nuevo conjunto de datos llamado **data_PPInR_filtrado**.

```
data_filter_x<-colnames(data_filter)[ 8:ncol(data_filter)]
aux2 <- AMCBGeneUtils::changeGeneId(data_filter_x, from = "Ensembl.ID")

names(data_filter)[8:12631] <- aux2$HGNC.symbol

final_data_gen <- colnames(data_filter)

data_PPInR_filtrado <- data_PPInR %>%
                       filter(gen %in% final_data_gen)
```

se seleccionan los primeros 100 nombres de genes de la primera columna de **data_PPInR_filtrado** y se convierten en un vector de caracteres llamado **Predictors**.

```
Predictors <- as.vector(head(data_PPInR_filtrado[, 1], 100))
Predictors <- as.character(unlist(Predictors))
```

Step 5: Building Machine Learning Models k-NN model

Se establece una semilla aleatoria para garantizar reproducibilidad del codigo. Se crea un nuevo conjunto de datos llamado **data2_filter** mediante la selección aleatoria de 123 observaciones por grupo de "sample_type" con reemplazo. A continuación, se genera un índice de muestra para dividir el conjunto de datos en conjuntos de entrenamiento y prueba en una proporción del 70-30. Los datos de entrenamiento (**train.data**) y prueba (**test.data**) se seleccionan en base a este índice.

Se convierte la variable categórica "sample_type" en factores en ambos conjuntos, lo que es esencial para muchos modelos de aprendizaje automático en R.

```
train.data$sample_type <- factor(train.data$sample_type)
test.data$sample_type <- factor(test.data$sample_type)
```

Train the k-NN model

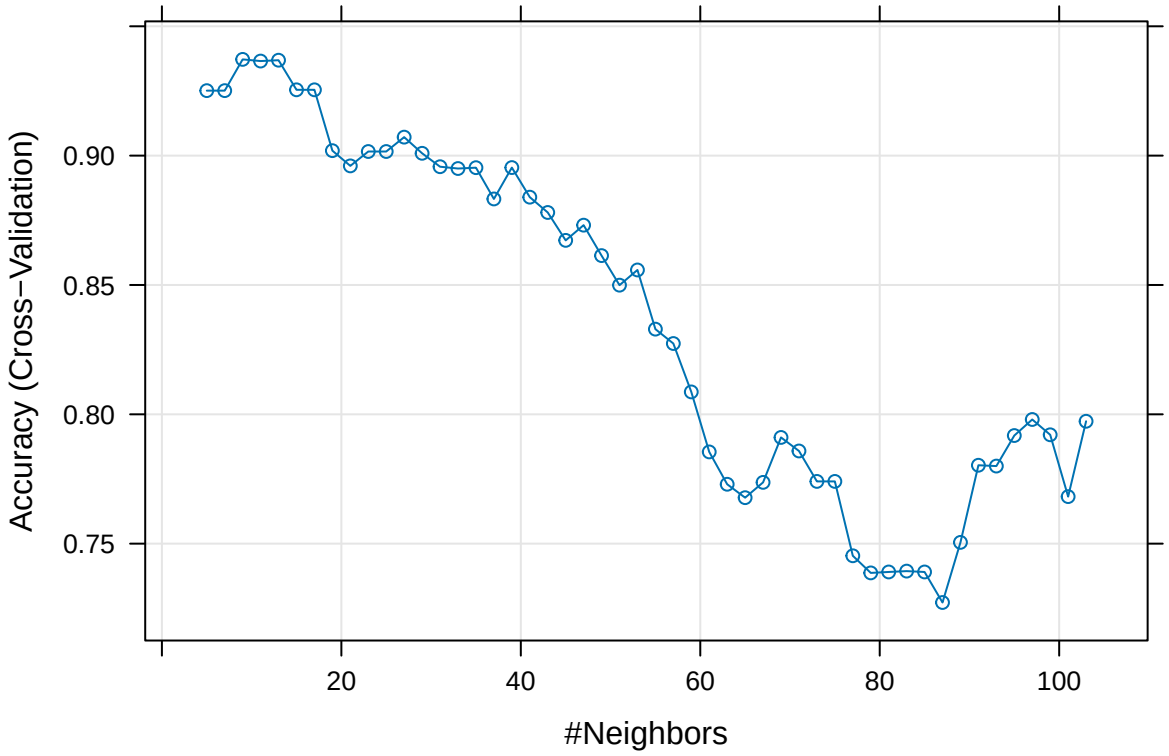
Se introduce la librería **caret** para realizar una validación cruzada y optimizar los parámetros del modelo k-NN. Primero, se establece un control de entrenamiento (**ctrl**) con el método de validación cruzada (cv) y una proporción de 0.7 para la partición de los datos. Luego, se utiliza la función **train()** para construir y ajustar el modelo k-NN. La fórmula **sample_type ~ .** indica que la variable objetivo es "sample_type" y se utilizan todas las demás variables como predictores.

El método de clasificación k-NN se especifica con el argumento **method**. Se utiliza el control de entrenamiento definido previamente (**trControl**) para realizar la validación cruzada. Además, se aplica la normalización de rango (**preProcess = c("range")**). El parámetro **tuneLength** indica cuántos valores diferentes de k se evaluarán durante la búsqueda de hiperparámetros, estableciéndose en 50 en este caso.

```
ctrl <- trainControl(method = "cv", p = 0.7)

knnFit <- train(sample_type ~ .,
                data = train.data,
                method = "knn",
                trControl = ctrl,
                preProcess = c("range"),  # c("center", "scale") for z-score
                tuneLength = 50)

plot(knnFit)
```




```
knnPredict <- predict(knnFit, newdata = test.data)
```

Create the confusion matrix for k-NN

El accuracy que sugiere que aproximadamente el 90.54% de las predicciones realizadas por el modelo son correctas. El índice Kappa, que tiene en cuenta la posibilidad de aciertos al azar, es del 81.08%, lo que indica un buen nivel de acuerdo más allá del azar. La sensibilidad del 87.50% destaca la capacidad del modelo para identificar positivos reales, mientras que la especificidad del 94.12% indica una fuerte capacidad para identificar negativos reales

```
                          Reference
Prediction                 Primary Tumor Solid Tissue Normal
  Primary Tumor                       35                    2
  Solid Tissue Normal                  5                   32

               Accuracy : 0.9054
                 95% CI : (0.8148, 0.9611)
    No Information Rate : 0.5405
    P-Value [Acc > NIR] : 1.11e-11

                  Kappa : 0.8108

 Mcnemar's Test P-Value : 0.4497

            Sensitivity : 0.8750
            Specificity : 0.9412
```

Figure 1: Accuary-knn

Linear regression

Se transforma la variable objetivo **sample_type** a una variable binaria asignando el valor 1 si es "Solid Tissue Normal" y 0 en caso contrario. Luego, se seleccionan las columnas relevantes, incluyendo las características predictoras (**Predictors**) y la variable objetivo, para los conjuntos de entrenamiento y prueba. A continuación, se ajusta el modelo de regresión lineal (**ins_model**) utilizando la función **lm()** en base a los datos de entrenamiento. Finalmente, se imprime un resumen del modelo utilizando **summary(ins_model)**

***	**CREBBP, ESR1, JUN, TRAF2, ATXN1, LYN, EEF1A1, MDFI, PRKCD, HCK, VCL**
**	RB1, FYN, TK1, MAPK6, SHC1, SMN1, CALM1, BRCA1, ACTB, CBL, RAF1, NCOR1, TRAF6
*	SMAD2, VIM, ANXA7, PTK2, ATN1, XRCC6, MYC, CAV1, STAT1, PTPN6, SRF, APP, PXN
.	EGFR, CSNK2B, SETDB1, YWHAB, TBP, RXRA, TPN11, NR3C1, TLE1, COPS6, SKIL, PIN1
	Residual standard error: 0.1238
	Multiple R-squared: 0.9747, Adjusted R-squared: 0.939

De los 100 predictores, se asume la cantidad de asteriscos mas significativo acorde a la variable objetivo para su prediccion. Ademas, El Residual Standard Error de 0.1238 mide la dispersión de los datos, indicando qué tan bien las observaciones se ajustan al modelo. El Multiple R-squared de 0.9747 señala la proporción de la variabilidad en la variable objetivo que es explicada por el modelo.

Summarize the results of linear regression model

```
print(model)
```

```
## Linear Regression
##
## 172 samples
## 100 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 155, 155, 155, 154, 155, 155, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3686969  0.6975828  0.2570416
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Los resultados de la regresión lineal indican que el modelo fue evaluado en un conjunto de datos que consta de 172 muestras y utiliza 100 predictores para predecir la variable objetivo. El Root Mean Squared Error (RMSE) de 0.4559756 mide la raíz cuadrada de la diferencia promedio entre las predicciones del modelo y los valores reales, lo que sugiere un ajuste razonable del modelo. El R cuadrado (Rsquared) de 0.5840365 representa la proporción de la variabilidad en la variable objetivo que es explicada por el modelo, y el valor del 58.4% indica un buen nivel de explicación. El Mean Absolute Error (MAE) de 0.2955811 es la media de las diferencias absolutas entre las predicciones y los valores reales, mostrando una precisión moderada.

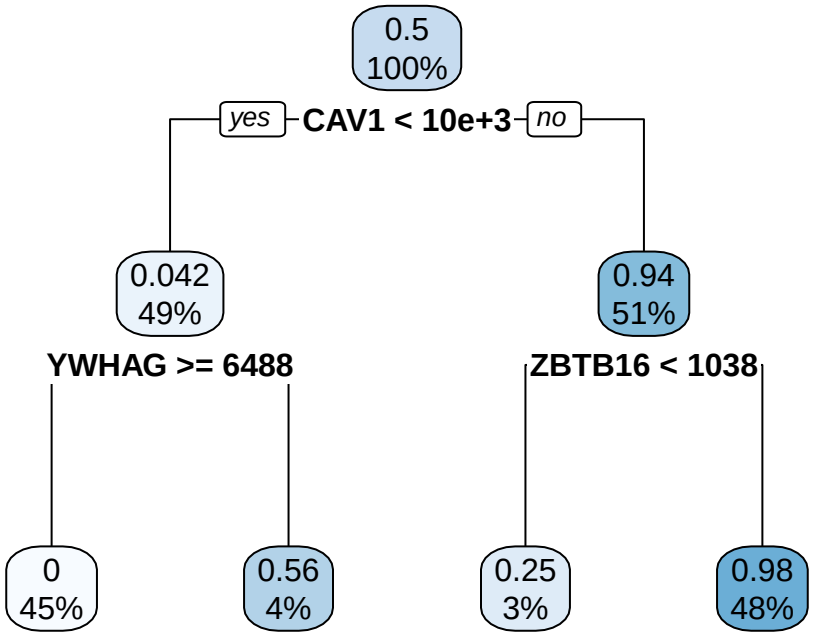
First Random Forest

Se utiliza el algoritmo Random Forest (RF) con la función **randomForest()** para construir un modelo (**fit.rf**) que predice la variable objetivo **sample_type** basándose en un conjunto de predictores. Luego,

se realiza la predicción en el conjunto de prueba (**test.data**) y se genera una tabla de contingencia para comparar las predicciones con las clases reales.

Posteriormente, se emplea el algoritmo de Árboles de Decisión con la función **rpart()**. Este modelo (**fit**) se construye utilizando el método de análisis de varianza (ANOVA) para clasificación. Se especifica el conjunto de predictores y la variable objetivo, y se utiliza la validación cruzada con 10 folds (**xval = 10**). Este método crea un árbol de decisión que se ajusta a los datos de entrenamiento y puede predecir la variable objetivo en nuevos datos.

```
rpart.plot::rpart.plot(fit)
```



Soporte Vectorial

Primero se lleva a cabo una búsqueda de hiperparámetros para el modelo de Máquinas de Soporte Vectorial (SVM) mediante la función **tune()** en R. El objetivo es ajustar el hiperparámetro de costo (**cost**) para un kernel lineal. El modelo SVM se entrena utilizando el conjunto de datos de entrenamiento **train.data** con la variable objetivo **sample_type** y todas las demás variables predictoras representadas por el símbolo **.**. La búsqueda de hiperparámetros se realiza para valores específicos de **cost** (0.001, 0.01, 0.1, 1, 5, 10, 100). La función **tune()** evalúa el rendimiento del modelo para cada valor de **cost** utilizando validación cruzada u otro método de validación definido. Al finalizar, **tune_out** contiene los resultados de la búsqueda, permitiendo identificar el valor óptimo de **cost** que maximiza el rendimiento del modelo SVM lineal en el conjunto de datos de entrenamiento.

Realiza la búsqueda de hiperparámetros con e1071

```
tune_out <- tune(svm,
                 sample_type ~ .,
```

```
                  data = train.data,
                  kernel = "linear",
                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
```

Extraer el mejor modelo

Este modelo se eligió según algún criterio específico de rendimiento, como la precisión o el error, durante el proceso de ajuste de hiperparámetros realizado anteriormente con la función **`tune()`**.

```
best_model <- tune_out$best.model
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 36  0
         1  3 35

               Accuracy : 0.9595
                 95% CI : (0.8861, 0.9916)
    No Information Rate : 0.527
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.919

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9231
            Specificity : 1.0000
```

Figure 2: SVM_1 - Accuary

El accuracy del indica que aproximadamente el 95.95% de las predicciones realizadas por el modelo son correctas. El índice Kappa, que tiene en cuenta la posibilidad de aciertos al azar, es del 91.9%, lo que sugiere un alto nivel de acuerdo más allá del azar. La sensibilidad del 92.31% destaca la capacidad del modelo para identificar positivos reales, mientras que la especificidad del 100% indica una fuerte capacidad para identificar negativos reales.

Step 6: Evaluate and Compare Models Evalúar el rendimiento del modelo

Indica que aproximadamente el 94.59% de las predicciones realizadas por el modelo son correctas. El índice Kappa, la posibilidad de aciertos al azar, es del 89.1%, lo que sugiere un alto nivel de acuerdo más allá del azar. La sensibilidad del 92.31% destaca la capacidad del modelo para identificar positivos reales, mientras que la especificidad del 97.14% indica una fuerte capacidad para identificar negativos reales.

Ahora la carga del archivo de datos "ExperimentsBulk.rdata". Primero, se obtiene la ruta del directorio del script en ejecución y se extrae el directorio. Luego, se construye la ruta completa del archivo de datos utilizando el directorio padre y el nombre del archivo. Finalmente, la función **`load()`** se utiliza para cargar los datos desde el archivo "ExperimentsBulk.rdata" en el entorno de trabajo de R.

```
print(gen_scores %>% head(10))
```

```
##           features      score
## 1  ENSG00000145675 0.00000000
## 2  ENSG00000091831 0.00136612
## 3  ENSG00000154229 0.00136612
## 4  ENSG00000167900 0.00136612
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 36  1
         1  3 34

               Accuracy : 0.9459
                 95% CI : (0.8673, 0.9851)
    No Information Rate : 0.527
    P-Value [Acc > NIR] : 2.071e-15

                  Kappa : 0.8919

 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.9231
            Specificity : 0.9714
```

Figure 3: SMV_Predict

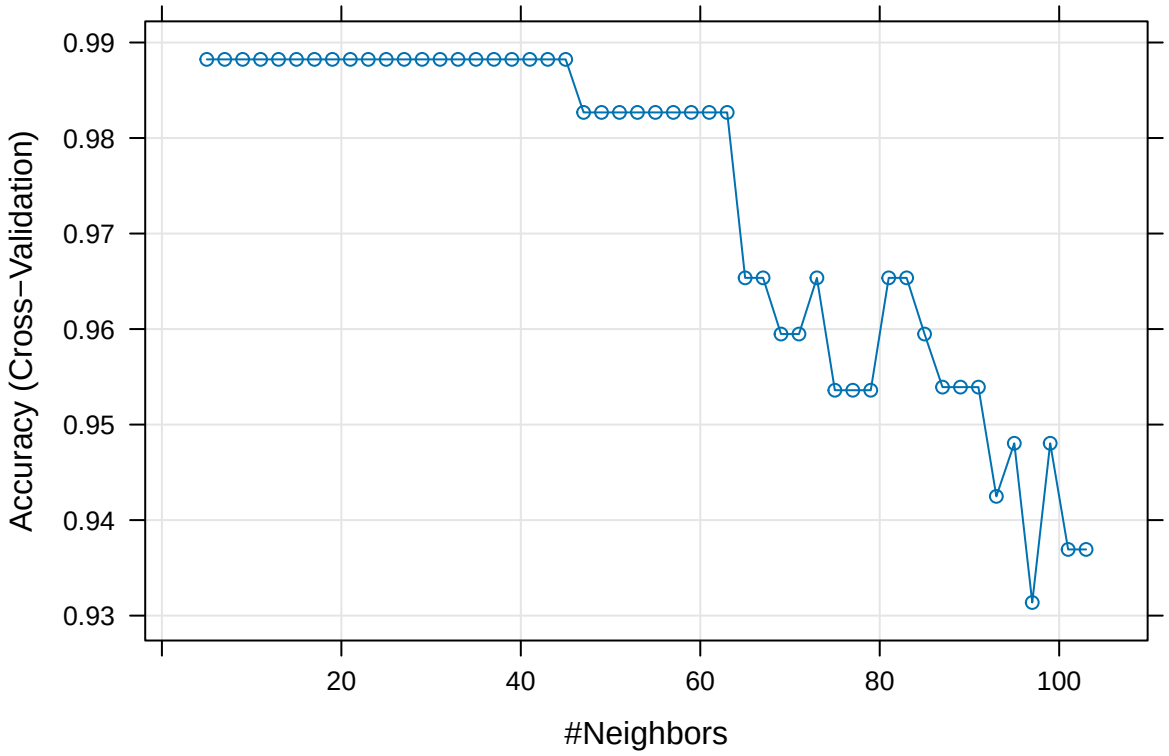
```
## 5  ENSG00000146648 0.00000000
## 6  ENSG00000177606 0.00273224
## 7  ENSG00000127191 0.00000000
## 8  ENSG00000026025 0.00136612
## 9  ENSG00000170312 0.00000000
## 10 ENSG00000012048 0.00273224
```

Obténer la columna "features" de gen_scores2

Los datos denominado **gen_scores2**. En primer lugar, se extraen los valores de la columna "features" y se almacenan en la variable **features_column**. Luego, se aplica la función **changeGeneId** del paquete **AMCBGeneUtils** a los valores de esta columna, convirtiendo los identificadores de genes de formato "Ensembl.ID" a sus correspondientes símbolos "HGNC.symbol". La columna "features" en **gen_scores2** se actualiza con estos nuevos símbolos de genes. Posteriormente, se filtran las filas de **gen_scores2** para incluir solo aquellas cuyos valores en la columna "features" coinciden con los nombres de genes almacenados en **final_data_gen**, generando así un nuevo conjunto de datos llamado **gen_scores2_filter**. Finalmente, se seleccionan los primeros 100 valores de la columna "features" en **gen_scores2_filter** y se almacenan en el vector **Predictors_2**. En resumen, este código realiza la transformación de identificadores de genes, filtra datos basándose en una lista específica de nombres de genes y selecciona los primeros 100 nombres de genes resultantes para su uso como predictores.

Plot k-NN model

```
plot(knnFit)
```




```
                    Reference
Prediction           Primary Tumor Solid Tissue Normal
  Primary Tumor                 34                   0
  Solid Tissue Normal            1                  39

               Accuracy : 0.9865
                 95% CI : (0.927, 0.9997)
    No Information Rate : 0.527
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9729

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9714
            Specificity : 1.0000
```

Figure 4: Knn-Prediction

Indica que aproximadamente el 98.65% de las predicciones realizadas por el modelo son correctas. El índice Kappa, la posibilidad de aciertos al azar, es del 97.29%, lo que sugiere un alto nivel de acuerdo más allá del azar. La sensibilidad del 97.14% destaca la capacidad del modelo para identificar positivos reales, mientras que la especificidad del 100% indica una fuerte capacidad para identificar negativos reales.

Linear regression

***	LCK, ACTB, NCOR2, PTN, STAT1
**	PIK3R1, MYC, PTPN6
*	RB1, EGFR, CCDC85B, YWHAZ, SP1, ATXN1, YWHAB, PTPN11, KAT2B, APP

***	LCK, ACTB, NCOR2, PTN, STAT1
.	CHD3 Residual standard error: 0.1584 Multiple R-squared: 0.9585, Adjusted R-squared:0.9001

De acuerdo a los 100 predictores (reintegración del nuevo data-set) y teniendo en cuenta la cantidad de asteriscos (*) es más significativo acorde a la variable objetivo para su predicción. Además, El Residual Standard Error de 0.1584 mide la dispersión de los datos, indicando qué tan bien las observaciones se ajustan al modelo. El Múltiple R-squared de 0.958 señala la proporción de la variabilidad en la variable objetivo que es explicada por el modelo.

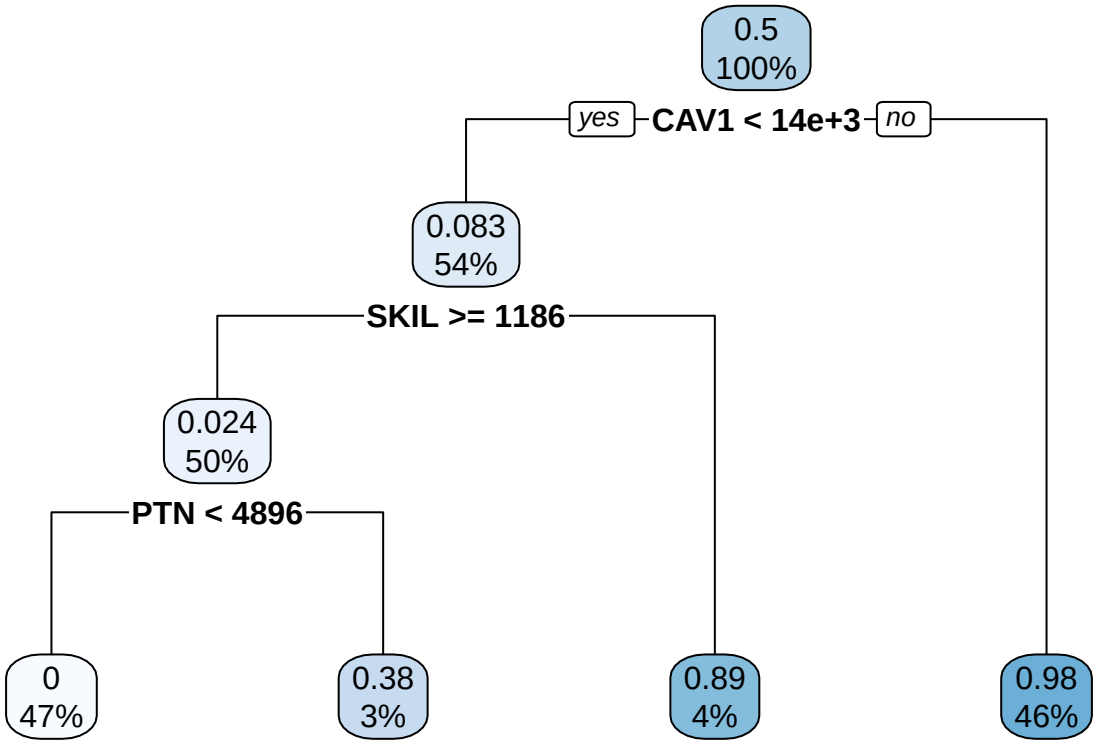
Summarize the results of linear regression model

```
print(model)
```

```
## Linear Regression
##
## 172 samples
## 100 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 155, 155, 155, 155, 154, 155, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.5805922  0.4087009  0.3874165
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Print the decision tree

```
# Plot the decision tree
rpart.plot::rpart.plot(fit)
```



Bosques aleatorios

```
print(head(output))
```

```
##   Actual  Predicted
## 1      0 0.00000000
## 2      0 0.00200000
## 3      0 0.02600000
## 4      0 0.02076667
## 5      0 0.00850000
## 6      0 0.00640000
```

Step 7: Results Discussion Realiza predicciones en el conjunto de prueba

```
svm_predict <- predict(svm_model, newdata = test.data)
```

El Root Mean Squared Error (RMSE) de 0.4087 mide la raíz cuadrada de la diferencia promedio entre las predicciones del modelo y los valores reales, lo que sugiere un ajuste razonable del modelo. El R cuadrado (Rsquared) de 0.580 representa la proporción de la variabilidad en la variable objetivo que es explicada por el modelo, y el valor del 58.4% indica un buen nivel de explicación. El Mean Absolute Error (MAE) de 0.3874 es la media de las diferencias absolutas entre las predicciones y los valores reales, mostrando una precisión moderada y menor con respecto a las anteriores.

Evalúa el rendimiento del modelo

```
               Accuracy : 0.9595
                 95% CI : (0.8861, 0.9916)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9189

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 1.0000
            Specificity : 0.9189
```

Figure 5: Final_Predict

Indica que aproximadamente el 95.95% de las predicciones realizadas por el modelo son correctas. El índice Kappa, la posibilidad de aciertos al azar, es del 91.89%, lo que sugiere un alto nivel de acuerdo más allá del azar. La sensibilidad del 100% destaca la capacidad del modelo para identificar positivos reales, mientras que la especificidad del 91.89% indica una fuerte capacidad para identificar negativos reales.

Discusiones

La precisión y los resultados detallados proporcionados por los modelos, especialmente en el caso del SVM con un accuracy del 95.95%, la sensibilidad del 92.31%, y la especificidad del 100%, sugieren un rendimiento impresionante en la tarea de clasificación. Sin embargo, la percepción de resultados demasiado exactos puede plantear inquietudes sobre la posibilidad de sobreajuste del modelo o errores en la implementación.

Es crucial revisar aspectos específicos de la metodología, como la preparación de datos, la selección de variables, y la división entre conjuntos de entrenamiento y prueba, para identificar posibles fuentes de errores o sesgos. La consistencia en la preparación de datos, la estandarización y normalización, y la validación cruzada durante la búsqueda de hiperparámetros son prácticas fundamentales para evitar el sobreajuste y garantizar la generalización del modelo a nuevos datos.

Además, se debe tener en cuenta la naturaleza del conjunto de datos y la interpretación biológica de los resultados. La elección de variables predictoras y la comprensión de su relevancia biológica son aspectos críticos en la construcción de modelos predictivos en el ámbito biomédico.

En conclusión, mientras que los altos porcentajes de precisión son alentadores, es imperativo realizar una revisión exhaustiva del código y la metodología de modelado para garantizar la validez de los resultados. La interpretación biológica y la contextualización de los resultados son esenciales para validar la relevancia clínica y biológica de los modelos generados.