Diego Jafet Garza Segovia - SVM y multiple testing

Dentro de este documento, se estara trabajando con la base de datos "Khan", el cual consiste de 83 muestras y 2308 variables de entrada, que consisten en la expresión génica estandarizada de distintos genes. La variable de salida cuenta con valores numéricos del 1 al 4 que corresponden a distintos tipos de cáncer.

Primeramente, tras importar la base de datos y revisar que no haya huecos (mediante las funciones read_csv, y isna().sum().sum() respectivamente), se separaran los datos en "X" y "y", con los datos pertenecientes a cada clase, y luego se calculara la diferencia del promedio entre las clases 2 y 4 para todos las genes.

In [50]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("A3.1 Khan.csv")

print("Dimensiones del dataframe:", df.shape)
df.head()

print("\nNumero de huecos:", df.isna().sum().sum())

X = df.drop(columns=["y"])
y = df["y"]

class1 = X[y == 1]
class2 = X[y == 2]
class3 = X[y == 3]
class4 = X[y == 4]

diff = (class2.mean() - class4.mean()).abs()
top10 = diff.sort_values(ascending=False).head(10)
print("\n", top10)
```

```
Dimensiones del dataframe: (83, 2309)

Numero de huecos: 0

 X187     3.323151
X509     2.906537
X2046    2.424515
X2050    2.401783
X129     2.165185
X1645    2.065460
X1319    2.045941
X1955    2.037340
X1003    2.011337
X246     1.837830
dtype: float64
```

En los resultados anteriores, se pudo confirmar que se importo adecuadamente la base de datos, que no hay huecos, y, que si hay multiples genes con una diferencia de promedio notable entre los de la clase 2 y la clase 4.

Estos 10 genes con la mayor diferencia de promedio varian desde aproximadamente 3.3 hasta 1.8. Normalmente, no habria forma de saber si esta es una gran diferencia o una pequeña diferencia entre las clases sin primero ver los valores promedios de manera individual por clase, sin embargo, ya que se especifica que esta base de datos esta "estandarizada", termina representando una GRAN diferencia. Por ejemplo, se puede decir que para el gen x187, hay 3.3 desviaciones estandares de diferencia entre la clase 2 y la clase 4.

En términos de inferencia estadística, diferencias tan marcadas sugieren que ciertos genes podrían ser altamente discriminativos entre tipos de cáncer. Si patrones similares se replican entre otras clases, entonces es razonable esperar que los genes con mayor variación entre grupos contribuyan fuertemente a diferenciar los distintos tipos de cáncer, ya que se mostraran genes particulares con un alto impacto en ciertos tipos de cancer. Esto deberia permitir tener una alta capacidad de inferencia.

Para poder comprobar que las diferencias de las medias de todos los genes de las clases 2 y 4 son significativas (y que no se cometio un error tipo I), se calculara el estadistico t y el p-value de los genes. Se calcularan mediante 3 metodos distintos, Bonferroni, Holm, y Benjamini-Hochberg. Esto nos permitira corregir por multiples pruebas e indicar para cada una cuales son los genes significativamente distintas entre las clases (manejando un control del 0.05).

Primero, la metodologia de Bonferroni consiste en controlar el family-wise error rate (FWER), es decir, disminuir la probabilidad de haber cometido un error tipo I al multiplicar el p-value por la cantidad de pruebas (m), rechazando las que sean menor a α (0.05). Esto hace menos probable que se rechaze una hipotesis, eliminando casi todos los falsos positivos pero tambien generando falsos negativos, especialmente considerando el contexto actual, en el que hay 2308 genes con muy pocas muestras por clase (total de 83).

Mientras tanto, la metodologia de Holm difiere en que consiste el umbral utilizado para rechazar la hipotesis depende de todos los p-values calculados ya ordenados de menor a mayor, segun su posicion. Donde este se rechaza si el p-value es menor al umbral decreciente. El umbral es calculado usando $\alpha/(m - j + 1)$, volviendose asi mas flexible para p-values grandes. Esto permite rechazar más hipótesis verdaderamente significativas sin perder control del FWER.

Finalmente, la metodologia de Benjamini-Hochberg consiste en tratar de controlar el false discovery rate (FDR). Esto se consigue nuevamente calculando los p-values y ordenandolos de menor a mayor en una lista, y luego aplicando la formula jq / m para calcular el umbral multiplicando, y rechazando todas las hipótesis con p-values menores o iguales. Este enfoque es menos estricto que Bonferroni y Holm, por lo que suele detectar un mayor número de genes significativos pero tambien garantizando que no mas del 5% de dichas pruebas, en promedio, son falsos positivos.

In [51]:
```
from scipy.stats import ttest_ind
from statsmodels.stats.multitest import multipletests

p_values = []
t_stats = []

for gene in X.columns:
    t, p = ttest_ind(X2[gene], X4[gene], equal_var=False)
    t_stats.append(t)
    p_values.append(p)

p_values = np.array(p_values)
t_stats = np.array(t_stats)

# Crear dataframe resultados base
results = pd.DataFrame({
    "gene": X.columns,
    "t_stat": t_stats,
    "p_value": p_values
})

# Bonferroni
reject_bonf, p_bonf, _, _ = multipletests(p_values, alpha=0.05, method='bonferroni'
results["p_bonf"] = p_bonf
results["sig_bonf"] = reject_bonf

# Holm
reject_holm, p_holm, _, _ = multipletests(p_values, alpha=0.05, method='holm')
results["p_holm"] = p_holm
results["sig_holm"] = reject_holm

# Benjamini-Hochberg
reject_bh, p_bh, _, _ = multipletests(p_values, alpha=0.05, method='fdr_bh')
results["p_bh"] = p_bh
results["sig_bh"] = reject_bh

bonf = results[results["sig_bonf"] == True].head(10)
```

```
print("(Bonferroni) Numero de genes significativos:", results[results["sig_bonf"] =
print("\n", bonf[["gene", "t_stat", "p_value", "p_bonf"]])

holm = results[results["sig_holm"] == True].head(10)
print("\n(Holm) Numero de genes significativos:", results[results["sig_holm"] == Tr
print("\n", holm[["gene", "t_stat", "p_value", "p_holm"]])

bh = results[results["sig_bh"] == True].head(10)
print("\n(Benjamin-Hochberg) Numero de genes significativos:", results[results["sig
print("\n", bh[["gene", "t_stat", "p_value", "p_bh"]])
```

```
(Bonferroni) Numero de genes significativos: 72

      gene     t_stat       p_value        p_bonf
1       X2  -6.900138  7.383962e-08  1.704218e-04
35     X36   5.610781  7.885432e-07  1.819958e-03
66     X67  -4.793322  1.413077e-05  3.261383e-02
128   X129  -8.412602  2.516574e-09  5.808252e-06
173   X174  -6.974367  5.603441e-09  1.293274e-05
186   X187 -12.229464  3.716887e-16  8.578576e-13
187   X188   4.741984  1.755499e-05  4.051692e-02
228   X229  -6.959156  6.507348e-09  1.501896e-05
245   X246  10.558828  1.537507e-14  3.548567e-11
250   X251  -4.965027  1.048420e-05  2.419754e-02

(Holm) Numero de genes significativos: 72

      gene     t_stat       p_value        p_holm
1       X2  -6.900138  7.383962e-08  1.682067e-04
35     X36   5.610781  7.885432e-07  1.786050e-03
66     X67  -4.793322  1.413077e-05  3.168120e-02
128   X129  -8.412602  2.516574e-09  5.760437e-06
173   X174  -6.974367  5.603441e-09  1.282067e-05
186   X187 -12.229464  3.716887e-16  8.574859e-13
187   X188   4.741984  1.755499e-05  3.930562e-02
228   X229  -6.959156  6.507348e-09  1.488230e-05
245   X246  10.558828  1.537507e-14  3.540879e-11
250   X251  -4.965027  1.048420e-05  2.358945e-02

(Benjamin-Hochberg) Numero de genes significativos: 296

      gene    t_stat       p_value          p_bh
1       X2 -6.900138  7.383962e-08  5.497479e-06
2       X3  4.255350  9.621623e-05  2.343457e-03
28     X29  3.452663  1.113505e-03  1.412072e-02
35     X36  5.610781  7.885432e-07  4.136268e-05
51     X52  3.802063  4.065804e-04  6.427312e-03
66     X67 -4.793322  1.413077e-05  4.867735e-04
79     X80  2.935551  5.491499e-03  4.400826e-02
88     X89  3.024116  4.137918e-03  3.745221e-02
118   X119  4.543027  3.329840e-05  9.606587e-04
128   X129 -8.412602  2.516574e-09  2.904126e-07
```

Segun los resultados, se puede ver que, como era de esperar, la metodologia de Benhamini-Hochberg tuvo la mayor cantidad de genes significativos, siendo casi 4 veces mas que las de Bonferroni y Holm.

Sin embargo, tambien se puede ver que muchos de estos genes (como X2, X36 y X67) aparecen en todas las metodologias, lo cual nos puede dar a entender que son las mas significativas de todas. Adicionalmente, podemos ver que estos valores tienden a tener un t_stat mas elevado, mostrando que tienen una mayor diferencia de la media entre estas dos clases.

Para poder hacer una comparacion de las 4 clases, y no unicamente de las clases 2 y 4, en vez de trabajar con el estadistico t, se utilizara una prueba de analisis de varianza (ANOVA), mediante el uso de la funcion f_oneway importado de scipy.stats.

In [52]:
```
from scipy.stats import f_oneway

F_stats = []
p_values = []

for gene in X.columns:
    F, p = f_oneway(class1[gene], class2[gene], class3[gene], class4[gene])
    F_stats.append(F)
    p_values.append(p)

F_stats = np.array(F_stats)
p_values = np.array(p_values)

# Crear dataframe base con resultados
anova_results = pd.DataFrame({
    "gene": X.columns,
    "F_stat": F_stats,
    "p_value": p_values
})

# Bonferroni
reject_bonf, p_bonf, _, _ = multipletests(p_values, alpha=0.05, method='bonferroni'
anova_results["p_bonf"] = p_bonf
anova_results["sig_bonf"] = reject_bonf

# Holm
reject_holm, p_holm, _, _ = multipletests(p_values, alpha=0.05, method='holm')
anova_results["p_holm"] = p_holm
anova_results["sig_holm"] = reject_holm

# Benjamini-Hochberg (FDR)
reject_bh, p_bh, _, _ = multipletests(p_values, alpha=0.05, method='fdr_bh')
anova_results["p_bh"] = p_bh
anova_results["sig_bh"] = reject_bh

bonf = anova_results[anova_results["sig_bonf"]].sort_values("p_bonf").head(10)
print("\n", "(Bonferroni) Numero de genes significativos:", anova_results[anova_res
print("\n", bonf[["gene", "F_stat", "p_value", "p_bonf"]])

holm = anova_results[anova_results["sig_holm"]].sort_values("p_holm").head(10)
print("\n", "(Holm) Numero de genes significativos:", anova_results[anova_results["
print("\n", holm[["gene", "F_stat", "p_value", "p_holm"]])
```

```
bh = anova_results[anova_results["sig_bh"]].sort_values("p_bh").head(10)
print("\n", "(Benhamin-Hochberg) Numero de genes significativos:", anova_results[an
print("\n", bh[["gene", "F_stat", "p_value", "p_bh"]])
```

```
 (Bonferroni) Numero de genes significativos: 404

        gene     F_stat       p_value        p_bonf
1954  X1955  84.364086  1.459035e-24  3.367454e-21
1388  X1389  83.817537  1.772751e-24  4.091510e-21
1002  X1003  77.795622  1.618988e-23  3.736625e-20
2049  X2050  69.230799  4.733702e-22  1.092539e-18
245    X246  68.414042  6.633722e-22  1.531063e-18
741    X742  65.572797  2.195548e-21  5.067325e-18
0        X1  59.118264  3.839240e-20  8.860966e-17
2161  X2162  56.987623  1.035143e-19  2.389109e-16
1953  X1954  55.419914  2.182635e-19  5.037522e-16
1644  X1645  54.768403  2.988392e-19  6.897208e-16

 (Holm) Numero de genes significativos: 412

        gene     F_stat       p_value        p_holm
1954  X1955  84.364086  1.459035e-24  3.367454e-21
1388  X1389  83.817537  1.772751e-24  4.089737e-21
1002  X1003  77.795622  1.618988e-23  3.733387e-20
2049  X2050  69.230799  4.733702e-22  1.091118e-18
245    X246  68.414042  6.633722e-22  1.528410e-18
741    X742  65.572797  2.195548e-21  5.056347e-18
0        X1  59.118264  3.839240e-20  8.837930e-17
2161  X2162  56.987623  1.035143e-19  2.381863e-16
1953  X1954  55.419914  2.182635e-19  5.020061e-16
1644  X1645  54.768403  2.988392e-19  6.870312e-16

 (Benhamin-Hochberg) Numero de genes significativos: 1162

        gene     F_stat       p_value          p_bh
1954  X1955  84.364086  1.459035e-24  2.045755e-21
1388  X1389  83.817537  1.772751e-24  2.045755e-21
1002  X1003  77.795622  1.618988e-23  1.245542e-20
2049  X2050  69.230799  4.733702e-22  2.731346e-19
245    X246  68.414042  6.633722e-22  3.062126e-19
741    X742  65.572797  2.195548e-21  8.445542e-19
0        X1  59.118264  3.839240e-20  1.265852e-17
2161  X2162  56.987623  1.035143e-19  2.986387e-17
1953  X1954  55.419914  2.182635e-19  5.597246e-17
186    X187  54.615724  3.217900e-19  6.751740e-17
```

Ahora que ya se estan comparando las 4 clases, podemos ver como el numero de genes significativos aumento significativamente, y que muchos de los genes desplegados ya no son los mismos (no necesariamente significa que sea porque ya no son significativos, sino que porque ahora hay mas que antes no se estaban considerando como significativos).

A partir del F_stat, podemos entender que hay una gran variabilidad de los genes a lo largo de las 4 clases, aunque no sabemos exactamente de cuanto o en que clases son las que mas

varia. Estos genes significativos nos permitira "filtrar" la base de datos para poder trabajar con los genes que tengan una mayor probabilidad de poder discriminar a las distintas clases.

Despues, se buscara separar los datos en entrenamiento y prueba (70% y 30% respectivamente) y construir y entrenar un modelo de SVM con tres kernels. Uno lineal, uno polinomial d eorden 3, y uno radial. Para reducir la cantidad de tiempo de procesamiento, no se hara uso de todas las variables, sino de las mas significativas segun el analisis previo.

ES IMPORTANTE CONSIDERAR que esto caé dentro de "fuga de datos", ya que estamos usando los resultados de datos que se basaron en toda la base de datos sin previa separacion. Sin embargo, por esta ocasion se obviara este detalle.

In [53]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

# Filtrar únicamente genes significativos por BH
significant_genes_df = anova_results[anova_results["sig_bh"] == True]

# Seleccionar 1/5 (20%) de esos genes
num_total = significant_genes_df.shape[0]
num_select = num_total // 5  # un quinto exacto

# Ordenar por p_bh (más significativos primero)
significant_genes_df = significant_genes_df.sort_values("p_bh")

# Selección final
selected_subset = significant_genes_df.head(num_select)
selected_genes = selected_subset["gene"].tolist()

print("Numero de genes seleccionados para SVM:", num_select)

# Subconjunto de X con solo esos genes
X_selected = X[selected_genes]

X_train, X_test, y_train, y_test = train_test_split(
    X_selected, y, test_size=0.3, random_state=42, stratify=y
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svm_linear = SVC(kernel="linear")
svm_linear.fit(X_train_scaled, y_train);

svm_poly = SVC(kernel="poly", degree=3)
svm_poly.fit(X_train_scaled, y_train);

svm_rbf = SVC(kernel="rbf")
svm_rbf.fit(X_train_scaled, y_train);
```

```
Numero de genes seleccionados para SVM: 232
```

Una vez generado los modelos de SVM, se calculara, para los 3 modelos, distintas metricas para poder comparar sus desempeños. Se compararan mediante matrizes de confusion, el cual nos permite ver cuantas predicciones fueron correctas e incorrectas (y como) por clase, la exactitud (el porcentaje total de aciertos sobre todas las clases), la precision (el porcentaje de predicciones hechas para una clase que fue correcto), la sensibilidad (la cantidad de muestras reales de una clase que fueron reconocidas correctamente), y el f1 score (la relacion entre la precision y el recall).

Las matrizes de confusion estaran ordenadas de las clases 1 a la 4 (izquierda a derecha y superior a inferior). El lado superior representa la clase verdadera y el lado izquierdo la clase predicha.

In [56]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

y_pred_linear = svm_linear.predict(X_test_scaled)
y_pred_poly   = svm_poly.predict(X_test_scaled)
y_pred_rbf    = svm_rbf.predict(X_test_scaled)

acc_linear = accuracy_score(y_test, y_pred_linear)
acc_poly   = accuracy_score(y_test, y_pred_poly)
acc_rbf    = accuracy_score(y_test, y_pred_rbf)

print("Matriz de confusión – Lineal")
print(confusion_matrix(y_test, y_pred_linear), "\n")

print("Matriz de confusión – Polinomial")
print(confusion_matrix(y_test, y_pred_poly), "\n")

print("Matriz de confusión – RBF")
print(confusion_matrix(y_test, y_pred_rbf), "\n")

def get_metrics(y_true, y_pred):
    report = classification_report(y_true, y_pred, output_dict=True)
    weighted = report["weighted avg"]

    return {
        "accuracy": accuracy_score(y_true, y_pred),
        "precision": weighted["precision"],
        "recall": weighted["recall"],
        "f1_score": weighted["f1-score"]
    }

# Obtener métricas de cada modelo
metrics_linear = get_metrics(y_test, y_pred_linear)
metrics_poly   = get_metrics(y_test, y_pred_poly)
metrics_rbf    = get_metrics(y_test, y_pred_rbf)

# Construir tabla final
comparison_table = pd.DataFrame([
    metrics_linear,
    metrics_poly,
    metrics_rbf
```

```
], index=["Lineal", "Polinomial", "RBF"])

# Imprimir tabla simple
print(comparison_table)
```

```
Matriz de confusión — Lineal
[[3 0 0 0]
 [0 9 0 0]
 [0 0 5 0]
 [0 0 0 8]]

Matriz de confusión — Polinomial
[[3 0 0 0]
 [0 9 0 0]
 [0 2 3 0]
 [0 1 0 7]]

Matriz de confusión — RBF
[[3 0 0 0]
 [0 9 0 0]
 [0 0 5 0]
 [0 0 0 8]]

            accuracy  precision  recall  f1_score
Lineal          1.00       1.00    1.00  1.000000
Polinomial      0.88       0.91    0.88  0.877238
RBF             1.00       1.00    1.00  1.000000
```

Se puede ver que en las matrizes de confusion, a excepcion de la clase 2 en el modelo Polinomial, se obtuvieron predicciones perfectas para cada clase. En particular, se puede notar que predijo erroneamente e veces que un cancer era de la clase 2 cuando era de la 3, y 1 vez cuando era de la clase 4. Esto nos podria indicar que para el modelo polinomial, el modelo sobreaprendio a identificar canceres como de la clase 2, causando equivocaciones. Es decir, pudo haber caido bajo algo de sobreajuste.

Tras ver las matrizes de confusion, como seria de esperar, la exactitud, precision, sensibilidad y el f1 score del modelo lineal y radial son perfectos. Mientras tanto, al modelo que "peor" le fue, fue al polinomial. Sin embargo, las metricas de desempeño resultantes de este modelo siguen siendo bastante buenos, siendo que el valor mas bajo que tiene es (aproximadamente) un 0.87.

Las posibles razones por las cuales los resultados salieron asi de positivos, podria ser debido a el numero de variables que habia con paracion con la cantidad de muestras. Mas aun, se seleccionaron previamente a genes extremadamente significativos (es decir, se hizo una fuga de datos), y mostraron una y otra vez que estos genes tenian diferencias de medias gigantes entre las clases, haciendolas facilmente discriminatorias.

Sin embargo, tambien se puede entender que simplemente, muchos de estos genes son verdaderamente utiles para lograr determinar de manera confiable si esta presente un tipo de cancer en particular o no.