Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Eléctrica

Tarea 3

SVM y Random Forest

Integrantes: Vincko Fabres
Profesor: Pablo Estevez.
Auxiliar: Ignacio Reyes Jainaga
Ayudantes: Andrés González
Bastián Andreas
Daniel Baeza
Francisca Cona
Javier Molina
Óscar Pimentel
Pablo Montero
Roberto Cholaky

Fecha de entrega: Sábado 13 de Noviembre de 2021
Santiago, Chile

1. Parte teórica

1.1. Imagine que está entrenando un modelo Random Forests. Dada una característica elegida para dividir el espacio ¿qué criterio se utiliza para colocar el umbral de clasificación? ¿Cómo se decide qué característica se utilizaría para dividir el conjunto de muestras en un nodo?.

El umbral de clasificación utiliza como criterio el indice de gini o entropía para ver la pureza de la partición. Para decidir la característica a escoger se selecciona la variable que dado su punto de corte disminuya la impureza.

1.2. En los modelos Random Forests cada árbol de decisión clasifica utilizando un subconjunto de las características disponibles. Explique por qué se hace esto.

Cada árbol de decisión tiene las siguientes características: bajo sesgo y alta varianza, razón por la cual una solución para el problema de clasificar es reducir las características que crean las particiones de esta forma se reduce la correlación entre los árboles bajando así la varianza.

1.3. ¿Qué son los métodos de kernel? ¿Qué ventaja tiene llevar los puntos a un espacio distinto? ¿Es necesario conocer la función que mapea los puntos al nuevo espacio para aplicar los métodos de kernel?

Los métodos kernel utilizan kernels las que son funciones que pueden verse como productos puntos generalizados, esto con la intención de aumentar la dimensionalidad y ahí encontrar el hiperplano óptimo. Al llevar los puntos a un espacio distinto; desde el espacio de entrada al espacio de características, creando un problema linealmente separable. Las funciones $\phi(x) * \phi(y)$ utilizadas para mapear al nuevo espacio no son necesarias conocerlas, ya que en su aplicación se utilizan directamente $k(x, y)$.

2. Parte práctica

Se deben resolver dos problemas de clasificación binaria utilizando los conjuntos de datos "two moons" y "Covertype dataset". Para resolver los problemas se utilizarán los clasificadores C-SVM y Random Forests.

2.1. Two Moons

Para este problema lo primero es importar las librerías necesarias, corriendo bloques de código dispuestos en el Jupyter Notebook.

Posterior a la importación de librerías se generan los conjuntos de entrenamiento y validación.

Una vez se tienen los conjuntos de deben comparar los casos de SVM lineal y SVM con kernel Gaussiano, calibrando los parámetros C y γ.

Una vez entrenado el SVM lineal calibrando C se obtiene lo siguiente:

	C=0.1	C=1	C=10	C=100
ROC validation	0.938404	0.939152	0.939180	0.939174
ROC train	0.933846	0.934180	0.934164	0.934156
SVs clase 1	195	172	169	169
SVs clase 2	195	172	170	169

Para el ajuste de γ y C en el SVM con kernel Gaussiano lo primero es fijar C en 1 y variar γ, dando los siguientes resultados:

	γ=0.1	γ=1	γ=10	γ=100
ROC validation	0.945680	0.981254	0.959800	0.952044
ROC train	0.941820	0.982462	0.972980	0.988140
SVs clase 1	186	107	127	328
SVs clase 2	186	107	124	343

Dados los desempeños anteriores variando γ se obtiene que el caso es el valor $\gamma = 1$, el cual es fijado para variar los valores de C, obteniendo lo siguiente:

	C=0.1	C=1	C=10	C=100
ROC validation	0.974864	0.981254	0.979506	0.974516
ROC train	0.974468	0.982462	0.983352	0.981488
SVs clase 1	177	107	86	81
SVs clase 2	175	107	85	83

Con estos datos se procede a graficar el AUC vs el valor del hiperparámetro, como se ve a continuación:

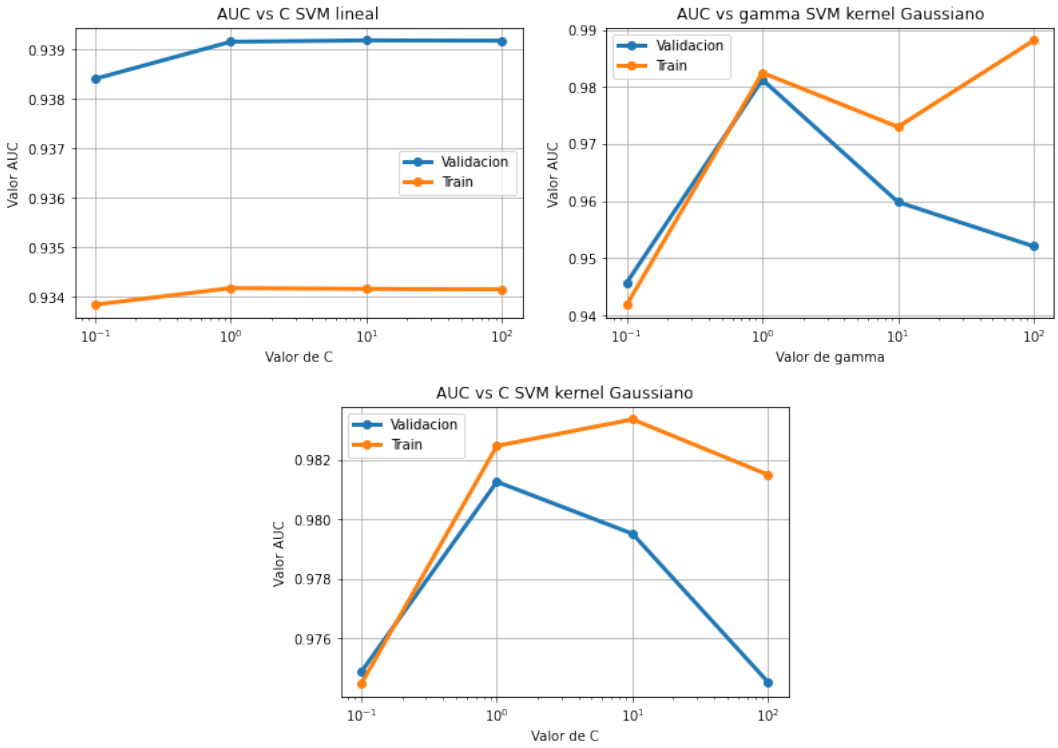

Figura 1: AUC vs parámetros de SVM

Escogiendo los mejores parámetros según AUC de conjunto de validación se obtiene como mejor SVM lineal el hiperparámetro C=1 y para SVM de kernel Gaussiano $\gamma = 1$ y C=1.
Los resultados de las fronteras de decisión se encuentran en Anexos, de los cuales se hace evidente el rol que tienen los kernels, pues al ser lineal la frontera es un hiperplano en el espacio de entrada, mientras que el kernel gaussiano realiza una partición no lineal, para ambos casos el hiperparametro C deforma levemente el hiperplano separador y disminuye el número de SVs, por otra parte el hiperparámetro γ al ser el inverso de radio de influencia de muestras al aumentar genera un hiperplano que aumenta la cantidad de SVs y 'condensa' los hiperplanos, generando nubes a modo de hiperplanos.

Se fija el número de árboles del modelo Random Forests a 1 para analizar cualitativamente, todas las imágenes generadas están en la sección de anexos 78.

Al observar la toma de decisiones del árbol al iniciar con profundidad 3 se ven decisiones simples, las cuales a medida aumenta su profundidad se complejizan mostrando un sobreajuste.

Se debe escoger la mejor combinación de parámetros para Randon Forests se número de árboles y profundidad máxima en 7, 50×3, 10. Utilizando la métrica AUC en el conjunto de validación se obtiene lo siguiente:

	depth=3	depth=10
# árboles =7	0.956514	0.964392
# árboles =50	0.963624	0.974426

Con lo cual la mejor tupla es 50 árboles con profundidad máxima 10.

2.2. Covertype dataset

El objetivo de esta parte es predecir la clase de vegetación en diversos lugares entre 7 clases existentes, sólo a partir de información cartográfica, proveyendo 54 características.

Lo primero es cargar la base de datos que contiene 581,012 observaciones del tipo de vegetación dominante en diversas áreas del Roosevelt National Forest en Colorado, USA.

Al cargar la base de datos es claro notar el desbalance de clases, donde la segunda clase posee 100 veces más muestras que la cuarta, siendo la mayor y menor cantidad de muestras respectivamente con 198310 y 1923 elementos en su haber.

Se puede lidiar con la problemática anteriormente mencionada mediante una técnica que asigan diferentes pesos a cada muestra de forma inteligente; asignandolos de forma inversamente proporcional a la frecuencia de la clase a la que pertenece. Al clasificar con y sin esta técnica los resultados son los siguientes:

	recall validación	recall train	# clasificados correctamente
Sin pesos	0.509	0.509	131795
Con pesos	0.805	0.815	115675

El número de ejemplos clasificados correctamente es mayor cuando no se utiliza esta técnica, lo cual se debe a un sobreajuste dado el desbalance de clases, razón por la cual se recomienda la configuración con asignación de pesos ya que si bien el número de clasificaciones correctas es menor su desempeño es mejor según las métricas.

Se debe utilizar el modelo Random Forest con correción al desbalance, midiendo qué valor de profundidad máxima entre [10,30,50] tiene mejor desempeño en conjunto de validación, los resultados son los siguientes:

	recall validación	recall train	# clasificados correctamente
Depth = 10	0.805	0.815	115675
Depth = 30	0.907	0.999	164835
Depth = 50	0.899	1.000	165706

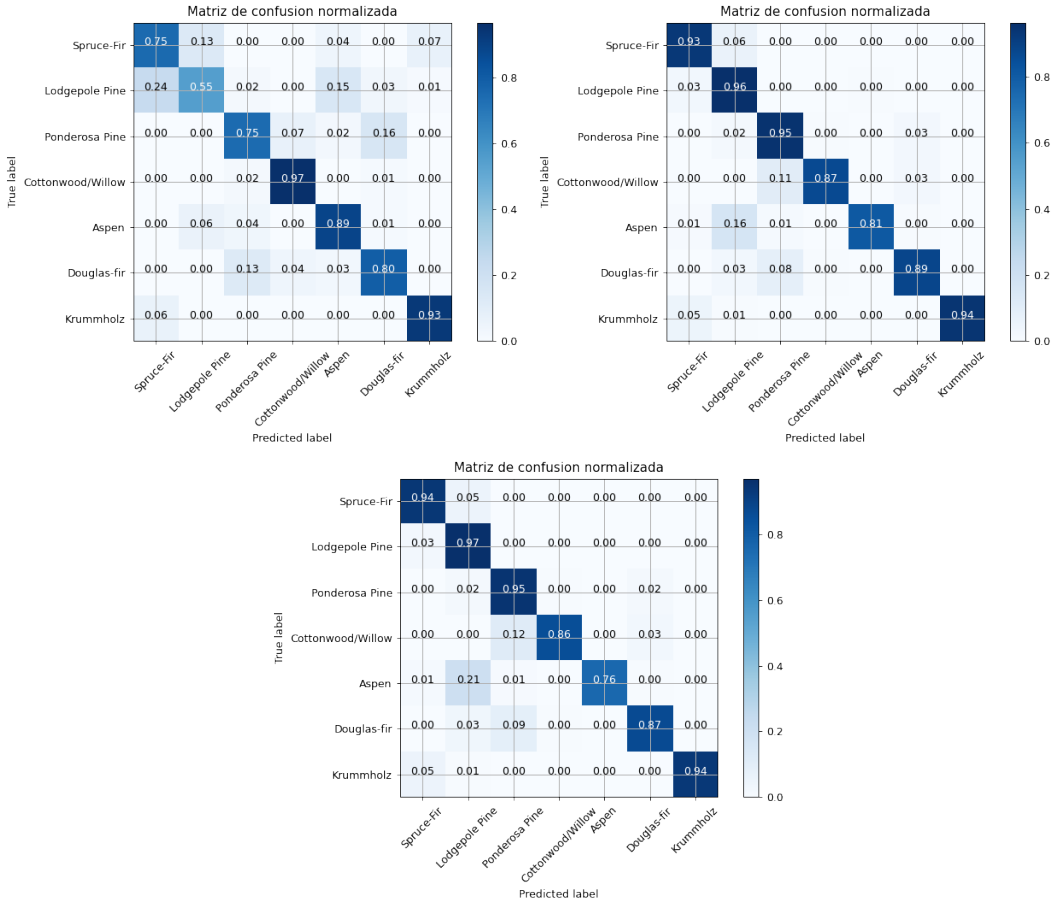

Figura 2: Matriz de confusión normalizada para Randoms Forests con corrección por desbalance

Con lo cual el mejor desempeño es el realizado por el modelo con profundidad máxima 30.

Con este modelo las mejores 5 características en orden descendente son: Elevation, Horizontal distance to roadways,Horizontal distance to fire points,Horizontal distance to hydrology y Vertical distance to hydrology. Para capturar el 80 % de la importancia se necesitan 12 características, es decir, el 22.22 % de las caracterísiticas.

3. Programación

Se debe construir un subconjunto del data set covertype con 1000 muestras por clase tomadas al azar, posteriormente particionar en conjuntos de entrenamiento y validación, normalizar caracteristicas y posteriormente entrenar un SVM del tipo one versus all de kernel gaussiano probando al menos 5 combinaciones de hiperparámetros.

Para la realización de lo anteriormente mencionado se utiliza la librería pandas y se aplica el siguiente código:

```
df= pd.DataFrame(dataset['data'],columns=feature_names)
df['target']= dataset['target']
reordenar = df.sample(frac=1,random_state=1).reset_index(drop=True)
clase_1=reordenar.loc[reordenar['target'] == 1]
clase_2=reordenar.loc[reordenar['target'] == 2]
clase_3=reordenar.loc[reordenar['target'] == 3]
clase_4=reordenar.loc[reordenar['target'] == 4]
clase_5=reordenar.loc[reordenar['target'] == 5]
clase_6=reordenar.loc[reordenar['target'] == 6]
clase_7=reordenar.loc[reordenar['target'] == 7]
new_covertype= pd.concat([clase_1[:1000],clase_2[:1000],clase_3[:1000],clase_4[:1000],clase_5
    ↪ [:1000],clase_6[:1000],clase_7[:1000]])
new_covertype=new_covertype.sample(frac=1,random_state=1).reset_index(drop=True)
```

Con lo que se crea un DataFrame como se solicita, 1000 elementos por clase al azar .

```

y = new_covertype['target']
X =  new_covertype.iloc[:,:54]
X_train, X_test, y_train, y_test = train_test_split( X, y,test_size=0.3, random_state=1, stratify=
    ↪  y)

from sklearn import preprocessing
scaler= preprocessing.StandardScaler()
scaler =scaler.fit(X_train) #Entrenamiento de standardscaler
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

lastclassifier1 = SVC(C=0.1, kernel='rbf', gamma=1.0, probability=True, decision_function_shape=
    ↪ 'ovr')
lastclassifier1.fit(X_train, y_train)
y_pred1 = lastclassifier1.predict_proba(X_test)
y_test = np.array(y_test)

pred1=[]
for elemento in y_pred1:
  pred1.append(np.where(elemento==elemento.max())[0][0]+1)
pred1
```

```
cm1 = confusion_matrix(y_test,pred1)
recall1 = mean_recall(cm1)
aciertos=0

for i in range(len(pred1)):
    if pred1[i] - y_test[i] == 0.0:
        aciertos+=1
    else:
      None


print(f'recall promedio {recall1}')
print(f'tasa de aciertos {aciertos/2100 }')

```

Posteriormente se particionan los conjuntos y se normalizan las caracteristicas usando el conjunto de entrenamiento para posteriormente entrenar el modelo.

Una vez realizado lo anterior, viendo la tasa de aciertos y recall promedio para el conjunto de validación se tienen los mejores resultados para el segundo clasificador, el cual tiene de hiperparámetros $\gamma = 1$ y C=1, con métricas de desempeño de recall promedio = 0.760952380952381 y tasa de aciertos = 0.7609523809523809. El gráfico de la matriz de confusión normalizada es la siguiente:

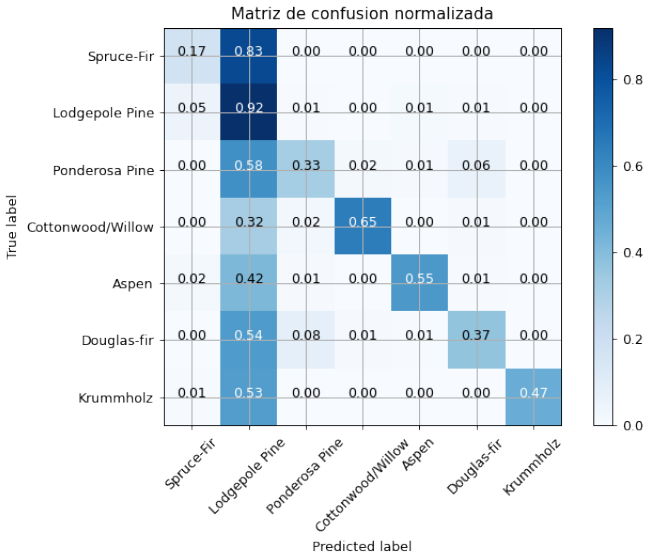

Figura 3: Matriz de confusión normalizada para mejor C-SVM kernel gaussiano para Covertype

Al comparar el desempeño de ambos clasificadores Random Forests clasifica mejor, esto se debe a que el SVM no es capaz de distinguir claramente la clase Lodgepole Pine, lo que baja enormemente su puntuación de desempeño.

4. Anexos

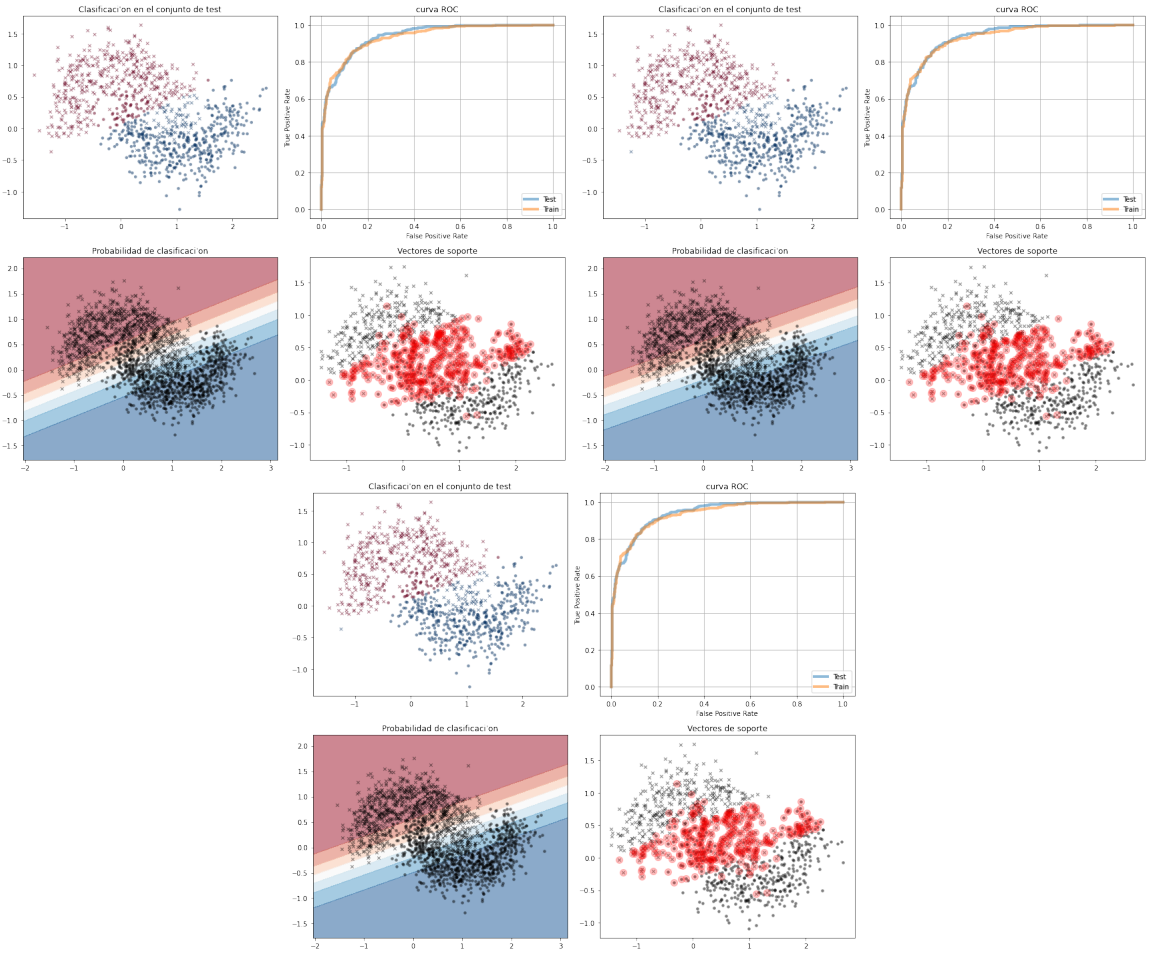

Figura 4: Plots SVM lineal

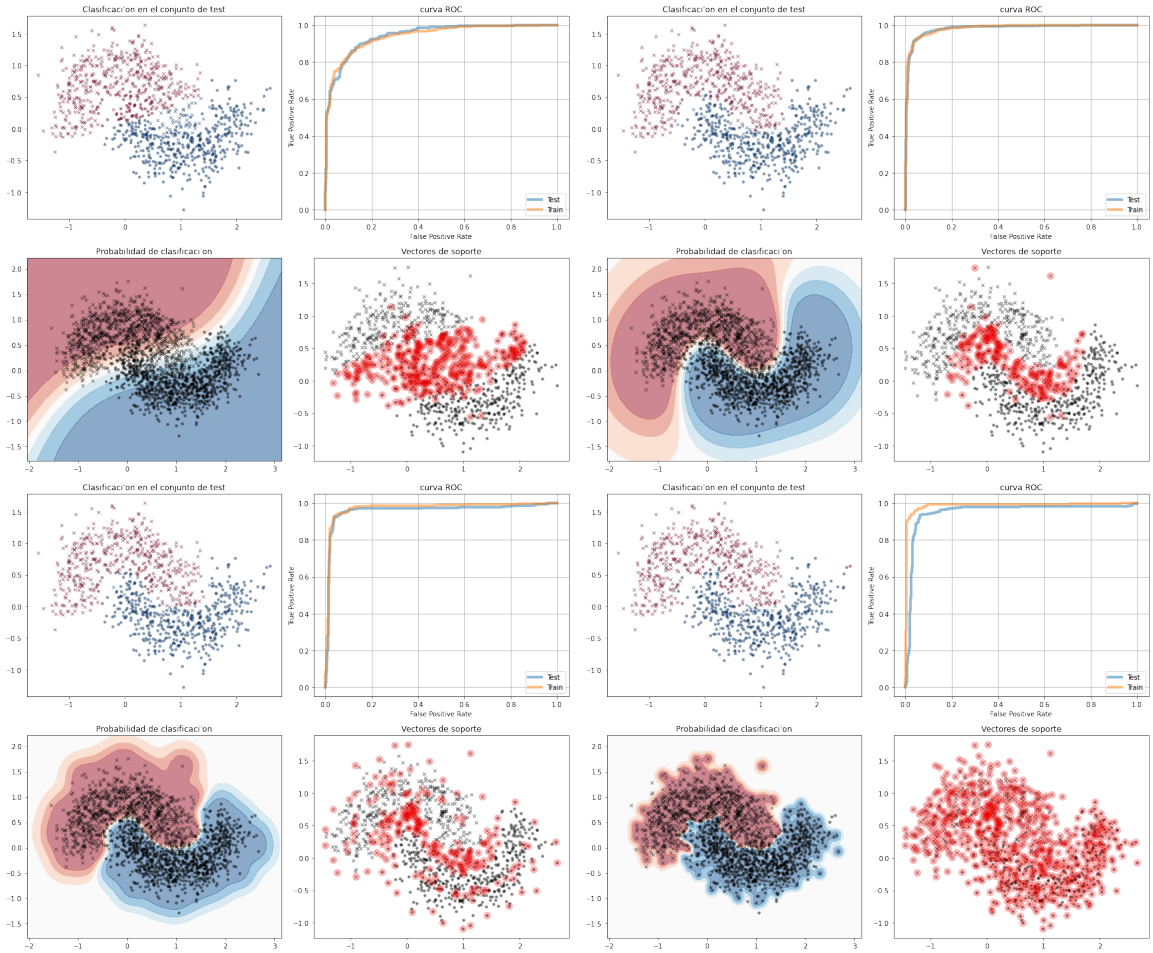

Figura 5: Plots SVM kernel Gaussiano variando gamma

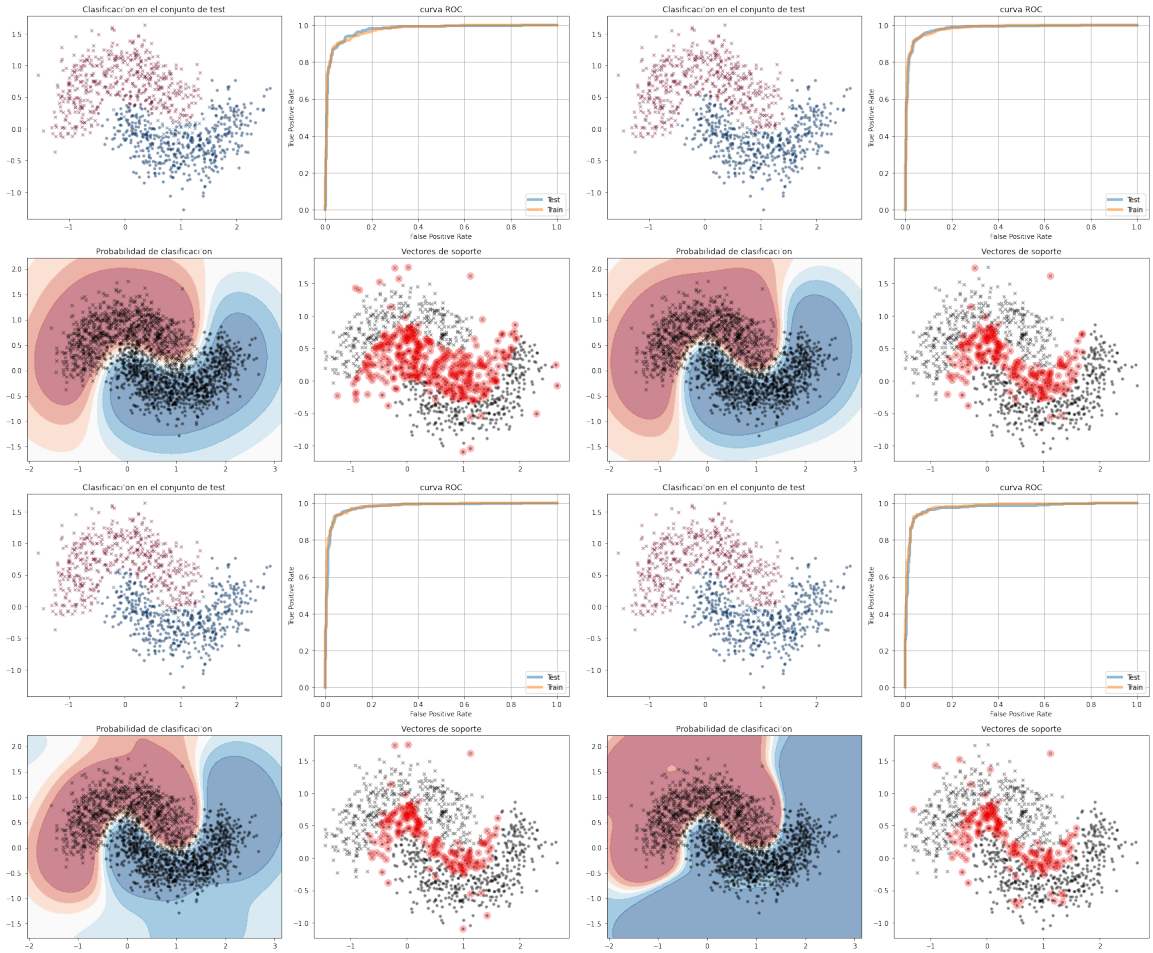

Figura 6: Plots SVM kernel Gaussiano variando gamma

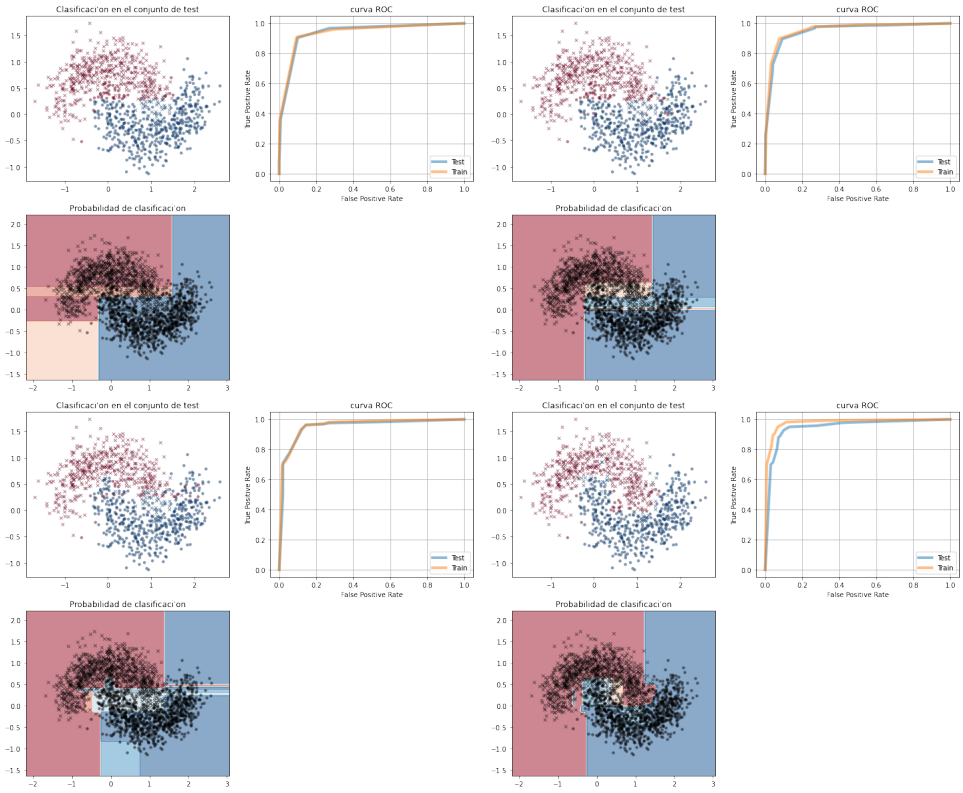

Figura 7: Plots Random Forest variando depth 3-6

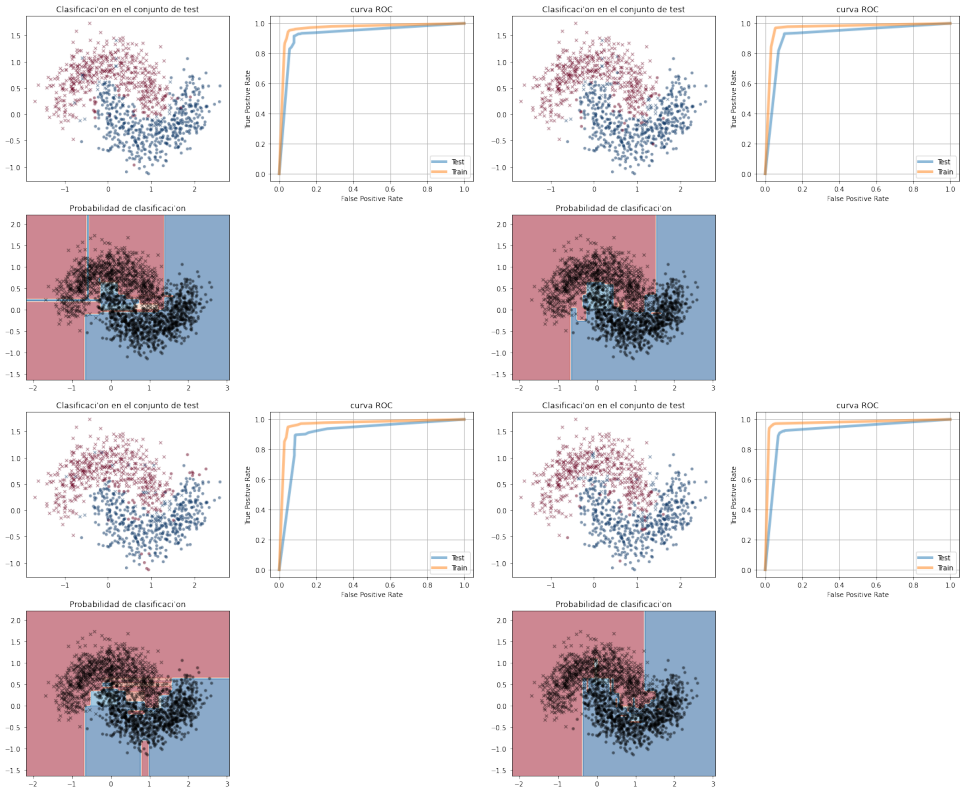

Figura 8: Plots Random Forest variando depth 7-10