Centro de Investigación en Matemáticas (CIMAT). Unidad Monterrey

Tarea 7

Hairo Ulises Miranda Belmonte

May 22, 2019

1 Problema

a) Aunque hay varias extensiones de AdaBoost al caso multiclase, una de las mas usadas es la llamada SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function), ya que está basada en la caracterización estadística de Friedman et al. Implementa esta versión de AdaBoost y verifi
ca su desempeño en un conjunto de datos con más de dos categorias.

Incluye una breve descripción del método basandote en el artículo: Zhu J, Zou H, Rosset S, and Hastie T (2009). Multi-Class AdaBoost. Statistics and Its Interface,2, 349360. Puedes usar también los datos que ahí se muestran para reproducir los resultados.

b) Usando los datos de los dígitos escritos a mano y digitalizados (mnist), complementa el ejercicio que hiciste en la tarea 2 aplicando métodos de clasi
cación basados en:

- Redes neuronales
- Máquinas de Soporte Vectorial
- Arboles de clasifi
  cación
- Random Forest
- AdaBoost

Utiliza K-Fold CV como criterio para elegir el mejor modelo, así como para compararlos. ¿ Qué método elegirías?

Especifica los parámetros que usaste en cada método de clasi
caciónn. Incluye gráfi
cos informativos sobre el desempeño de cada método. Actualiza tu aplicación interactiva, si es que la implementaste en la primera tarea.

c) Repite el ejercicio 2 para los datos de frutas que usaste en la tarea 4. Utiliza la representación en el espacio HSV con la mediana y los cuartiles centrales.

d) Puntos extra: verifica el desempeño en del clasificador que elegiste en ejemplos reales. Toma algunas fotos de frutas y realiza un preprocesamiento básico para clasi
carlas. Puedes usar el código en C (cortesía de Karen) para quitar el fondo de tu foto. Lee las instrucciones que vienen documentadas.

1.1 Solución inciso "a"

Explicación del algoritmo SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function)

Es un algoritmo boosting multiclase que utiliza una función de perdida exponencial; i.e, la función de costo es más costosa cuando tiende a errar en la clasificación, y caso contrario cuando acierta en la clasificación.

A diferencia del algoritmo AdaBoost, no necesariamente el clasificador débil predice el 50% de las veces de forma correcta, teniendo como consecuencia la introducción de más peso en las observaciones mal clasificadas. Otra diferencia es el factor $log(K-1)$ que se suma en el caso de tener más de dos clases, y como consecuencia hace que este algoritmo sea equivalente a ajustar un modelo aditivo utilizando una función de perdida exponencial.

A manera intuitiva el algoritmo realiza cierto número de iteraciones, que son los números de clasificadores débiles implementados, después calcula una suma ponderada por los pesos de las observaciones mal clasificadas, para obtener el valor de la función de perdida y de ahí actualizar los pesos con el fin de que el próximo clasificador débil contemple las observaciones mal clasificadas.

En este ejercicio se implementa la extensión multiclase AdaBoost, SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function). Se evalúa el desempeño del algoritmo al contrastarlo con AdaBoost M1, el cual trata los problemas multiclase como casos binarios.[1] El conjunto de datos que se utiliza se simulan como en el artículo de Zhu H, et al. (2009), que consiste en un problema de tres clases con 21 variables.

Sea x_j con tres clases, donde:

[1] Se utiliza la librería **ada**, y calcula el error de entrenamiento.

$$\begin{aligned}\text{clase 1: } & u * v_1(j) + (1-u) * v_2(j) + \epsilon_j \\ \text{clase 2: } & u * v_1(j) + (1-u) * v_3(j) + \epsilon_j \\ \text{clase 3: } & u * v_2(j) + (1-u) * v_3(j) + \epsilon_j\end{aligned}$$

con $j = 1, ..., 21$, u es una uniforme entre cero y uno, ϵ_j variable con distribución normal estándar, v_l formas de las ondas, con $v_1(j) = max(6 - |j - 11|, 0)$, $v_2(j) = v_1(j - 4)$, y $v_3(j) = v(j + 4)$.

En total se tiene un conjunto de datos de 300 observaciones, 100 por cada clase, de los cuales se toma el 80% como datos de entrenamiento y el resto como de prueba; cabe mencionar que el desempeño en los clasificadores solo se evalúa para los datos de entrenamiento.

En la figura 1.1, se presenta el error de entrenamiento que se obtiene al utilizar Adaboost.M1 y la extensión multiclase SAMME con un parámetro de complejidad del 0.01, se puede observar que a medida que incrementa el número de iteraciones, i.e, el número clasificadores débiles (árboles)[2], el error de entrenamiento de AdaBoost.M1, tiende a una constante; por otro lado, la tasa de error de entrenamiento al utilizar el método SAMME, después de varias iteraciones continua disminuyendo.

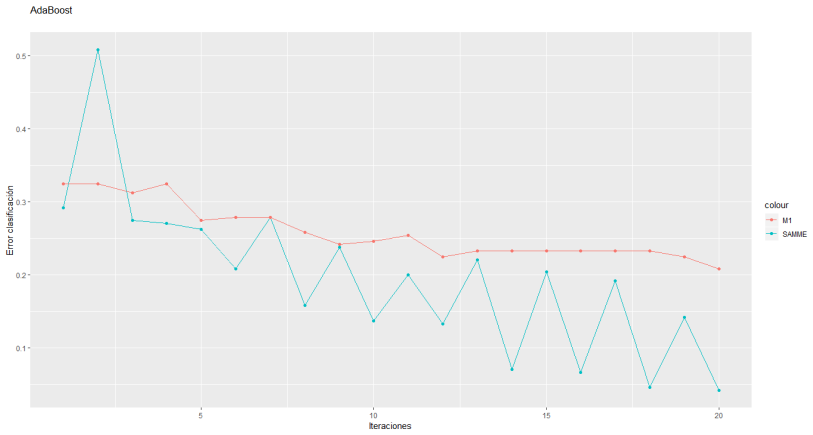

Figure 1.1: Error de entrenamiento, SAMME vs AdaBoost.M1

Siguiendo el artículo de Zhu H, et al. (2009), se toma un conjunto de datos de UC-Irvine machine learning archive, en especifico *netTest data*, donde la variable respuesta tiene dos clases, (0,1) , y el con un número de covariables de dos [3].Se calcula el error de clasificación para los datos de entrenamiento, y se observa la complejidad variando el numero de iteraciones; i.e, números de clasificadores (árboles)[4] débiles utilizados.
Para la clasificación se utiliza adaBoost.M1, el método SAMME, y el algoritmo SAMME implementado en R por la librería **adabag**, en la función **boosting**, cabe mencionar que se utiliza un parámetro de complejidad del 0.05 para los tres métodos.

[2]Se utiliza un árbol de decisión para clasificar, utilizando la función **rpart** en R, con un split máximo de cinco

[3]Se es consiente que estos clasificadores son para problemas multiclase, pero sólo para realizar más comparaciones con el algoritmo SAMME programado se realizan estos ejemplos

[4]Se utilizan dos *splits* máximos

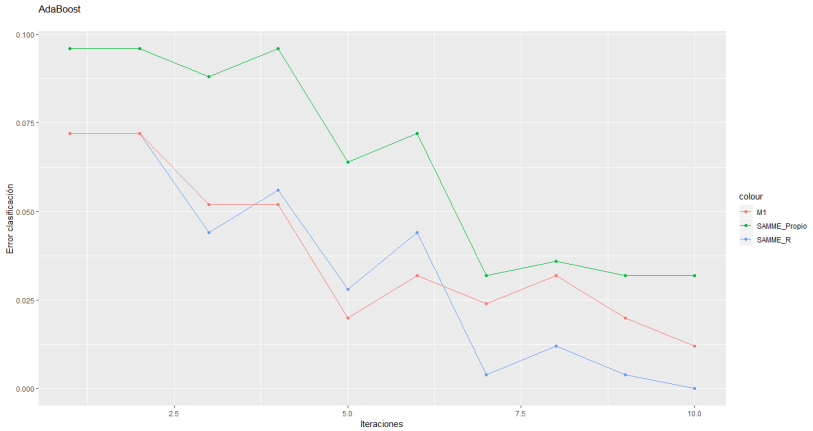

Figure 1.2: Error de entrenamiento, SAMME, SAMME.R y AdaBoost.M1

Entre los resultados se observa que el error de entrenamiento para los tres métodos sigue la misma tendencia; sin embargo, el método SAMME implementado con la librería de R, registra errores de entrenamiento menores al método SAMME que se implementa para este trabajo, esta diferencia es solo de un factor a escala; por otro lado, el erro de entrenamiento del método AdaBoost.M1 tiende a crecer conforme incrementa el número de iteraciones.
En conclusión SAMME presenta mejores resultados a AdaBoost.M1, lo cual implica que es mejor que los métodos multiclase que tratan el problema como casos binarios.

1.2 Solución inciso "b"

Se utilizan lose datos de dígitos escritos a mano y digitalizados (mnist), de la tarea dos. Se realiza un *append* a las etiquetas, datos de entrenamiento y prueba, con el fin de utilizar solo 10% de las observaciones, dado a la presencia de problemas computacionales.

El subconjunto de datos consta con 785 variables (contando las etiquetas), correspondientes a los pixeles de una imagen de $28x28$; se mantiene un total de 1600 imágenes, y se toma el 80% del subconjunto para entrenar los clasificadores (1280 imágenes), y 20% para probar su desempeño (320 imágenes).

Se evalúa el desempeño de los clasificadores por medio de *K.fold cross validation*, con $K = 5$; i.e, se divide en 5 el conjunto de entrenamiento y para cada subconjunto se ajusta un clasificador, evaluando su desempeño con los datos de prueba y calculando el error de clasificación para cada k, al final estos errores son promediados y sirven para la selección de los parámetros del modelo.

K-fold Cross Validation, se implementa para determinar la complejidad en cada clasificador, i.e., seleccionar los parámetros de SVM, RF, AD, RN, AdaBoost; una vez teniendo los clasificadores se vuelve a implementar K-fold Cross Validation para determinar que clasificador clasifica mejor a los dígitos.

Máquinas de Soporte Vectorial

Se hace uso de la librería **e1071** en R, y la función **svm**; se utiliza k-folds CV, para determinar la complejidad del clasificador, implementando SVM con un kernel gaussiano con distintos valores para el parámetro del kernel y el de regularización (costo de mal-clasificar).

En la figura 1.3, se observa el error de clasificación medido con K-fold CV/5, para distintos valores de gamma (constante del kernel gaussiano), y distintos valores en el parámetro de complejidad. La figura inferior derecha muestra las gráficas en un solo cuadro, se observa que el error de prueba más pequeño se alcanza con un parámetro para el kernel gaussiano de 0.03, y un valor del parámetro de complejidad de 1.

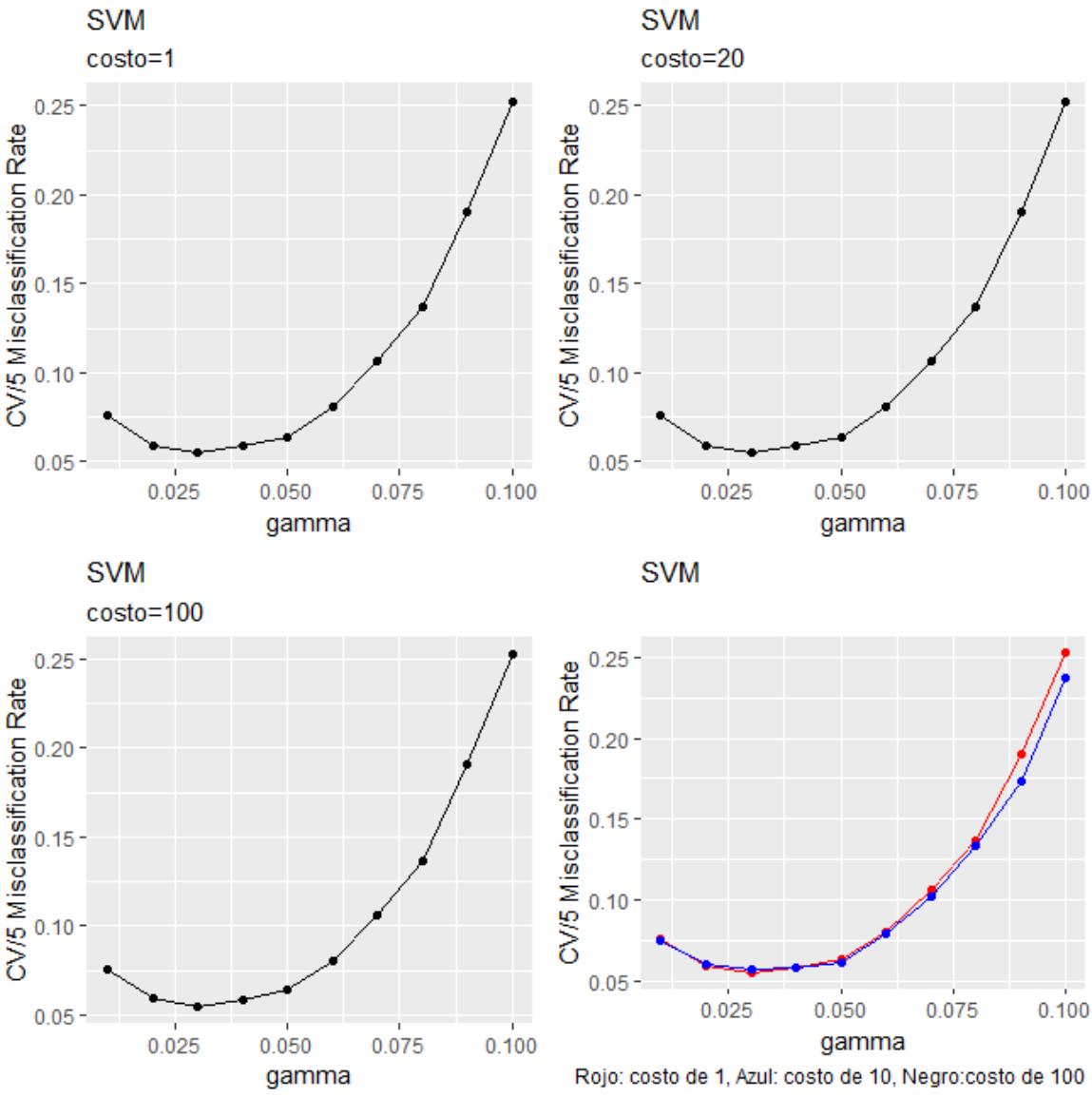

Figure 1.3: SVM K-fold error de clasificación para datos de prueba

Arboles de clasificación

Al igual que en SVM, se realiza CV con $k = 5$, variando los parámetros del árbol de decisión, los cuales son; tamaño máximo del árbol y el parámetro de complejidad (cp). En la figura 1.4, se observan los valores del error en la clasificación calculados con k-folds CV, se tiene que el parámetro de complejidad no proporciona información; de esta manera, se toma un árbol con altura máxima de 6 *splits*, valor donde el gráfico cambia a ser constante y con ello un error constante.

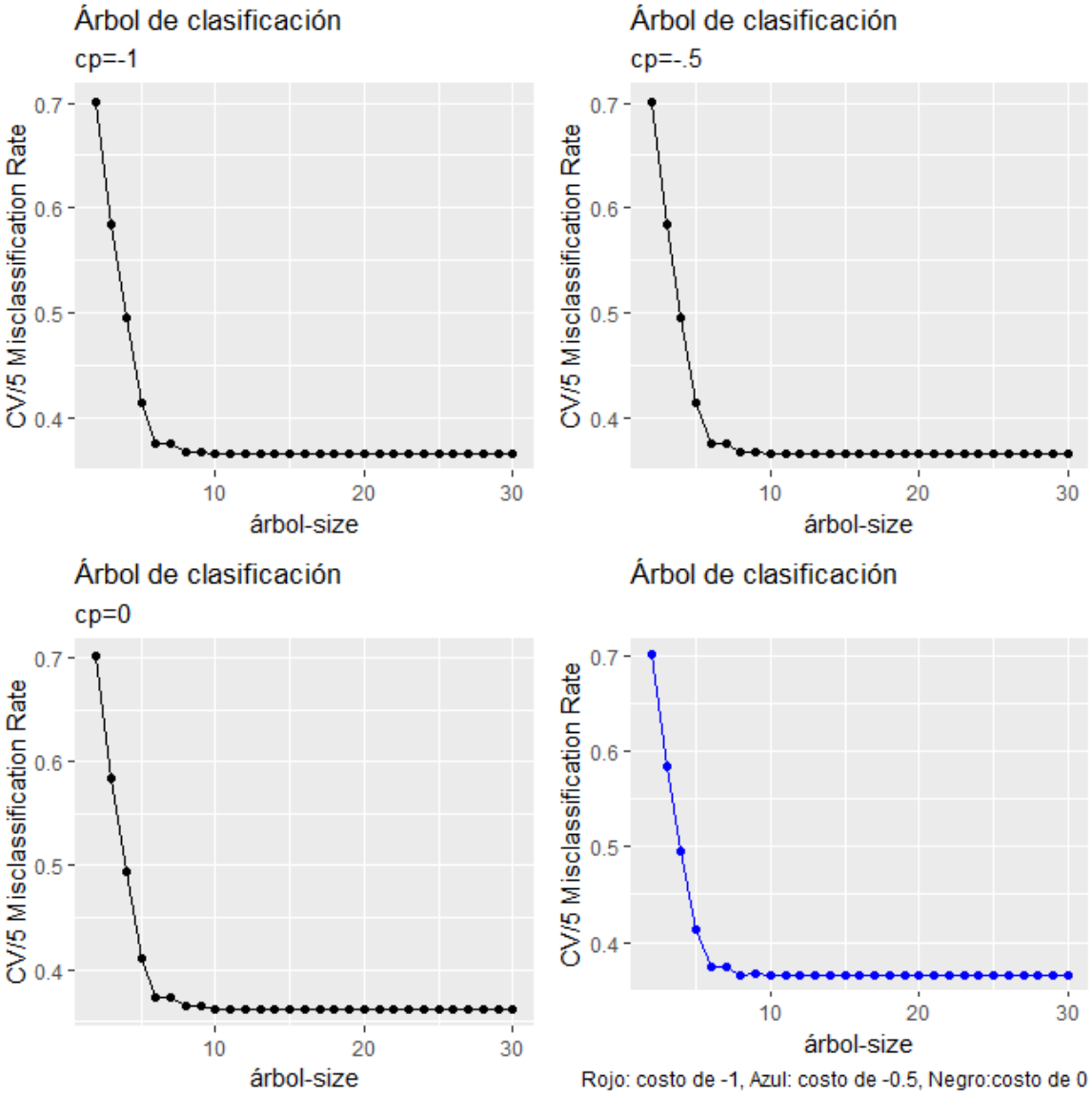

Figure 1.4: Árbol de clasificación K-fold error de clasificación para datos de prueba

Random Forest

Se realiza lo anterior para Random Forest, la figura 1.5, muestra el error de clasificación utilizando k-fold CV, con distinto números de variables y de árboles (iteraciones). Se puede observar en el gráfico inferior derecho, donde se contrastan las tres curvas, que utilizar 314 variables produce un error de prueba menor, y realizar o hacer uso de 200 árboles, proporciona el menor error de clasificación para los datos de prueba.

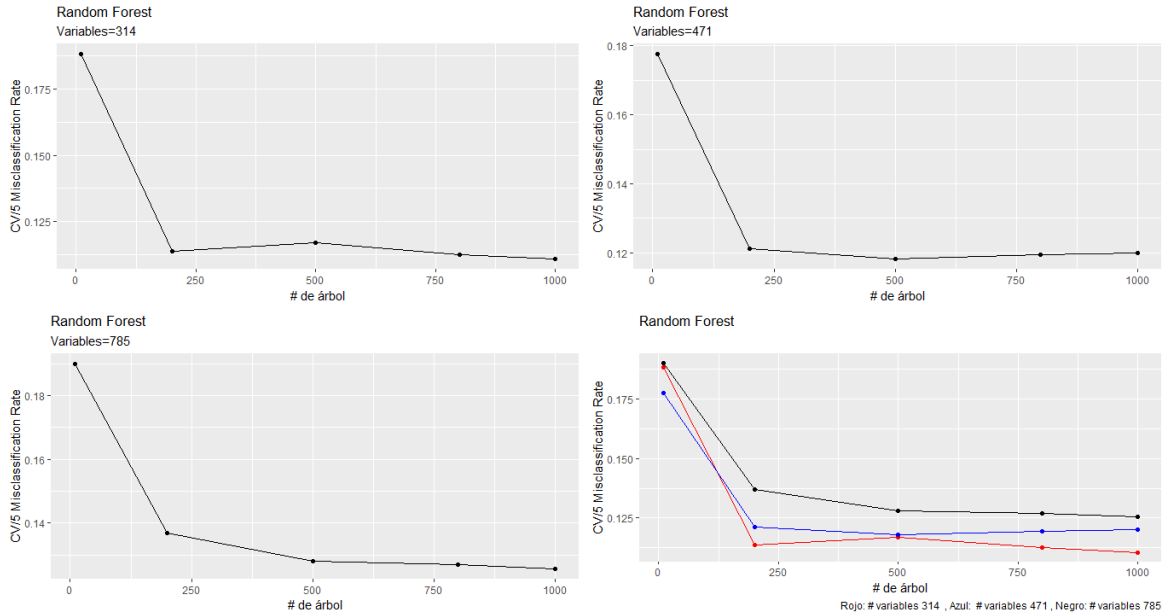

Figure 1.5: Random Forest K-fold error de clasificación para datos de prueba

AdaBoost

En la figura 1.6, se presenta el error de clasificación calculado con k-fold CV/5, al utilizar adaBoost con distintos números de iteraciones; i.e, número de clasificadores débiles (árbol de clasificación con máximo dos *splits*); asimismo cabe mencionar que se utiliza la extensión multiclase SAMME, propuesto por Zhu J (2006). Se observa que evaluando la complejidad del clasificador el de menor error de prueba es aquel que utiliza 20 iteraciones o clasificadores débiles (árboles de decisión).

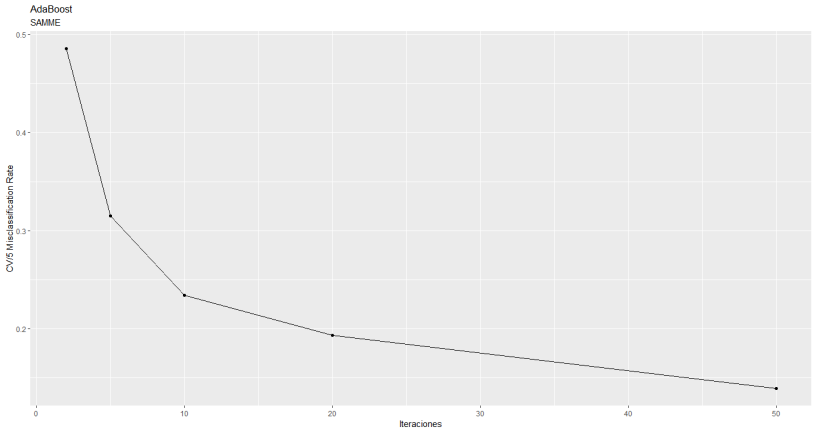

Figure 1.6: AdaBoost K-fold error de clasificación para datos de prueba

Redes Neuronales

Por problemas computacionales la selección del número de nodos en una red neuronal -con una capa oculta- no está dada por k-fold CV, para seleccionar la mejor arquitectura se utilizan los criterios AIC, BIC, y la *cross entropy error*. En la figura 1.7, se observa que la red con mayor AIC y BIC, pero menor error, es aquella que cuenta con 7 nodos ocultos, siendo este la arquitectura de la red que se contrastará respecto a los otros clasificadores.

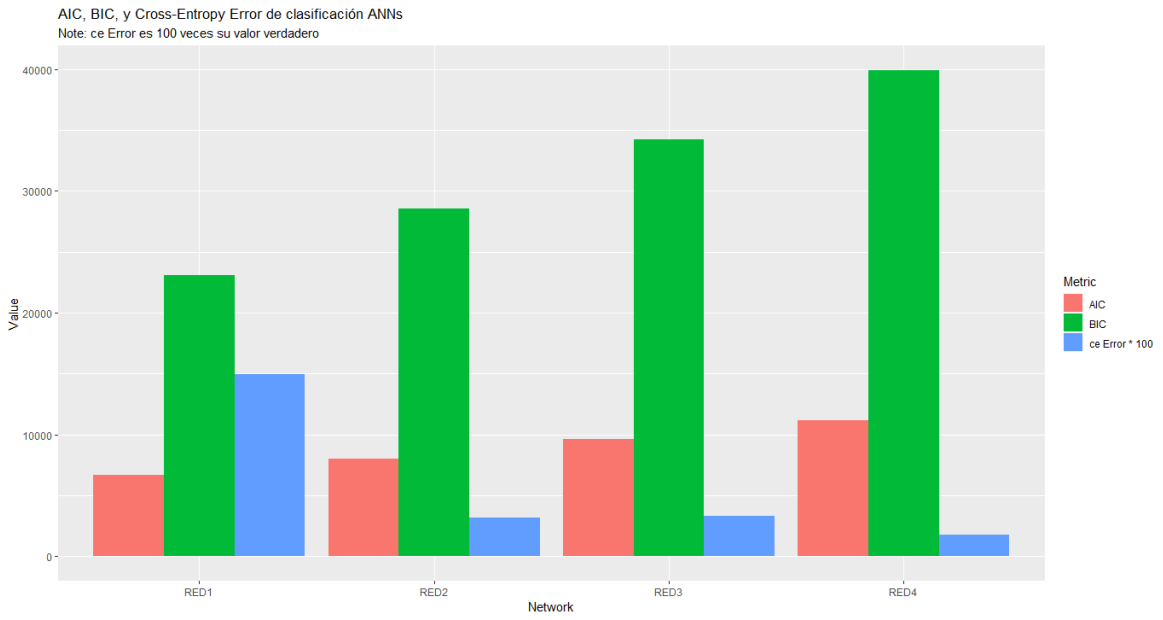

Figure 1.7: Red Neuronal K-fold error de clasificación para datos de prueba

Recapitulando lo anterior los clasificadores que se seleccionan son los siguientes; SVM con un kernel gaussiano con constante de 0.03, y parámetro de complejidad de uno; árbol de decisión con parámetro de complejidad de menos uno y altura máxima de 6 (*split*); random forest

indicando que realice boostrap sobre 314 variables y utilizando 200 árboles de clasificación débiles con dos **splits** como máximo; addaBoost caso multiclase SAMME, con 20 iteraciones (árboles con dos *splits* máximo); red neuronal con una capa oculta y 7 nodos ocultos.

Para los clasificadores anteriores en la tabla 1.1, se muestra el error de prueba calculado con k-folds cross validatios, con $k = 5$. Se observa que el clasificador con menor error en clasificación es el SVM con tasa de 8.3%, seguido de Random Forest con 11.8. Los de peor desempeño fueron el árbol de clasificación con 36.4%, y la red neuronal con 30.5%.

	SVM	Árbol clasificador	Random Forest	AdaBoost	Red Neuronal
Precisión	91.7%	63.6%	88.2%	80.7%	69.5%
Error	8.3%	36.4%	11.8%	19.3%	30.5%

Table 1.1: K-fold CV/5; Desempeño de los clasificadores con datos de prueba

En conclusión el método que se selecciona en base a k-fold CV, para la tarea de clasificación de dígitos es SVM con lo parámetros previamente mencionados, seguido de estese contemplaría el método Random Forest.

Para visualizar la clasificación con SVM, en la figura 1.8, se utiliza PCA para reducir dimensiones en los datos de entrenamiento y de prueba, las cruces rojas son los scores de los datos de prueba, los cuales fueron proyectados a las dos primeras componentes. En la figura 1.9, se implementa el clasificador SVM previamente mencionado, con el cual se entrena con los scores de los datos de entrenamiento y se prueban con los scores de los datos de prueba, se observa que las fronteras de clasificación tiende a separar lo mejor posible los dígitos y el error de prueba es de 0.128.

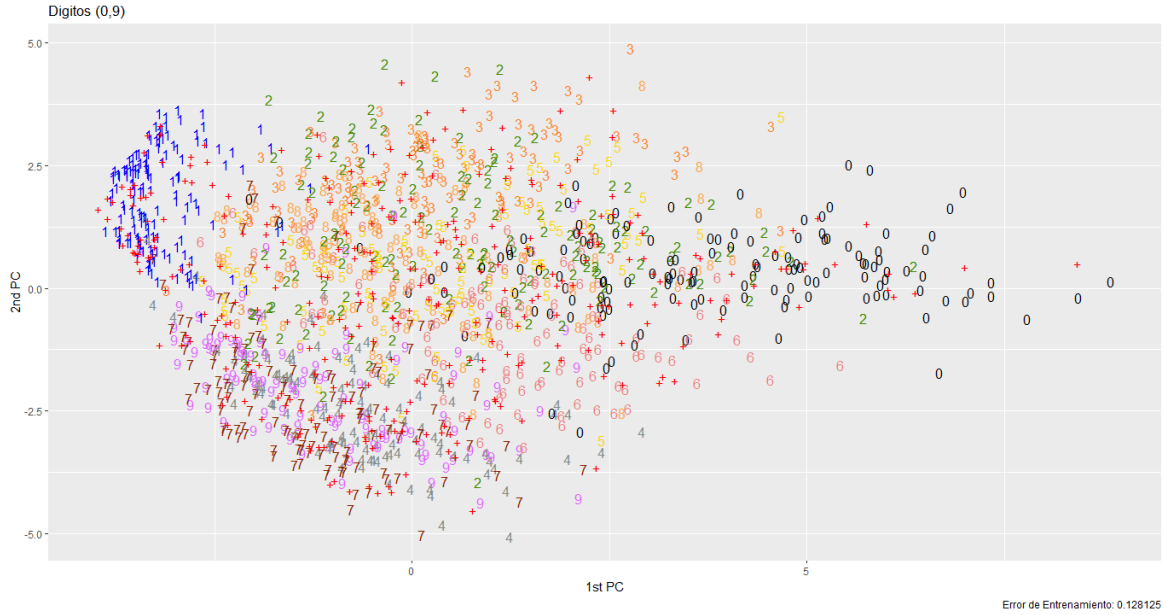

Figure 1.8: Clasificación con SVM

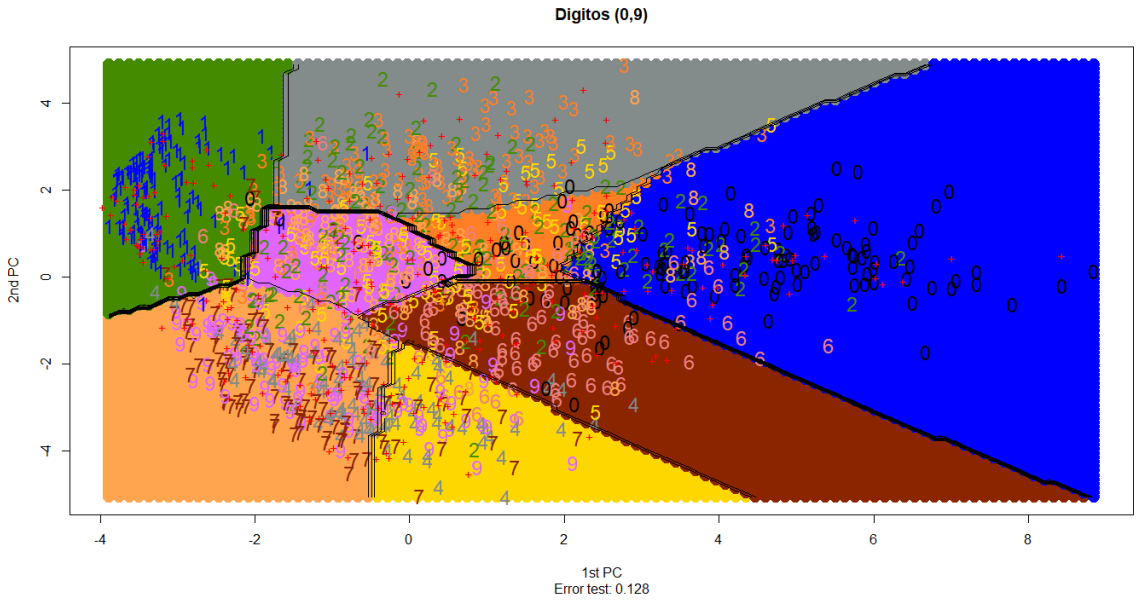

Figure 1.9: Clasificación con SVM; frontera de clasificación

Dígitos

Se actualiza la aplicación interactiva con el clasificador SVM previamente mencionado, los resultados en la clasificación de dígitos escritos son sin duda alguna mejores que los que se tenían. El archivo con el nombre "Digitos" contiene la aplicación interactiva que queda al lector el comprobar sus resultados, en este trabajo solo se ponen un para de imágenes del desempeño del clasificador.

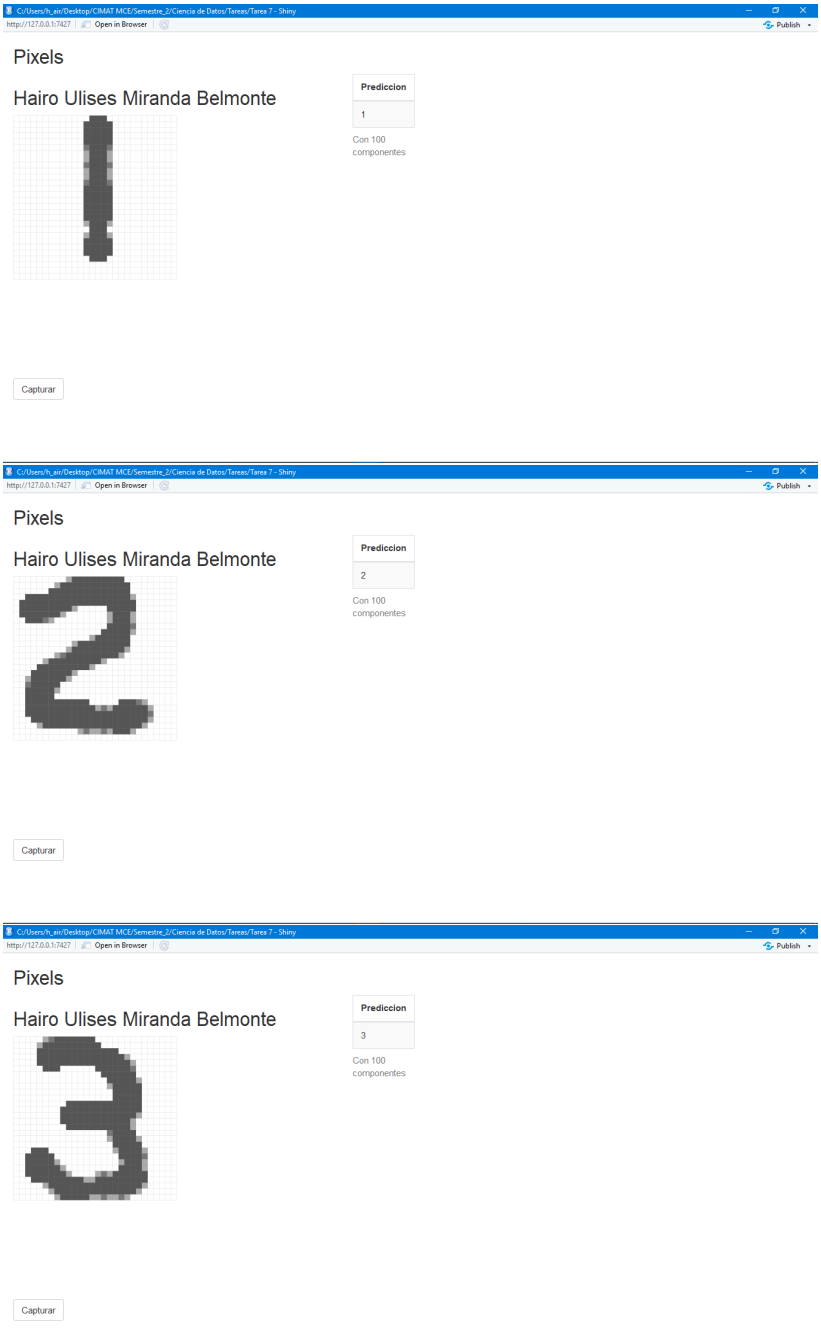

Figure 1.10: SVM clasificador

1.3 Solución inciso "c"

[5]

Canales HSV[6]

Se repite lo realizado en el ejercicio 2 para los datos de frutas que se utilizaron en la tarea 4. La imágenes son representadas con la mediana de los canales en el espacio HSV; se trabaja 1300 imágenes y tres canales (variables); el 80% de los datos se utilizan como conjunto de entrenamiento, el resto como datos de prueba.

En la figura 1.11, se presenta el error de prueba medido con k-folde cross validatios ($k = 5$), al implementar SVM con un kernel gaussiano y varios el valor para el termino constante. El menor error de clasificación para los datos de prueba ocurre cuando el parámetro del kernel tiene valor de siete.

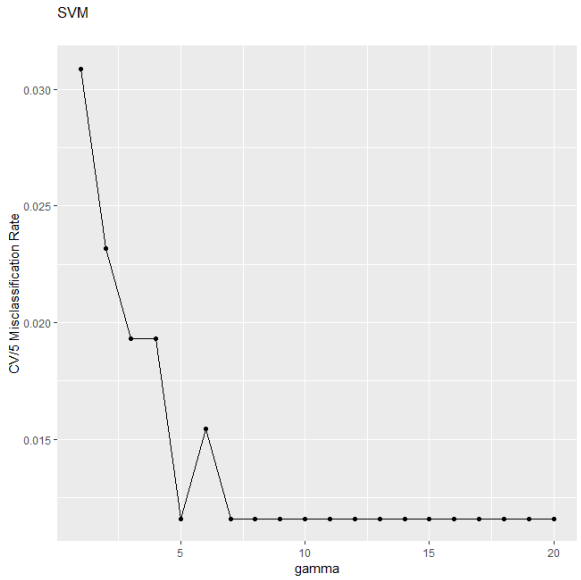

Figure 1.11: SVM K-fold error de clasificación para datos de prueba

En la figura 1.12, se observa el error de clasificación para datos de prueba al utilizar un árbol de decisión, el error se calcula con k-fold CV ($k = 5$), y variando sobre el tamaño del árbol. El menor error ocurre con un árbol cuya partición máxima es de siete.

[5]Nota: en caso de utilizar un valor para el parámetro de complejidad será indicado de manera explicita, en caso de no hacerlo el lector debe dar por hecho que no se utilizó alguno.

[6]No se utilizan la Redes Neuronales dado a problemas en el ajuste del clasificador

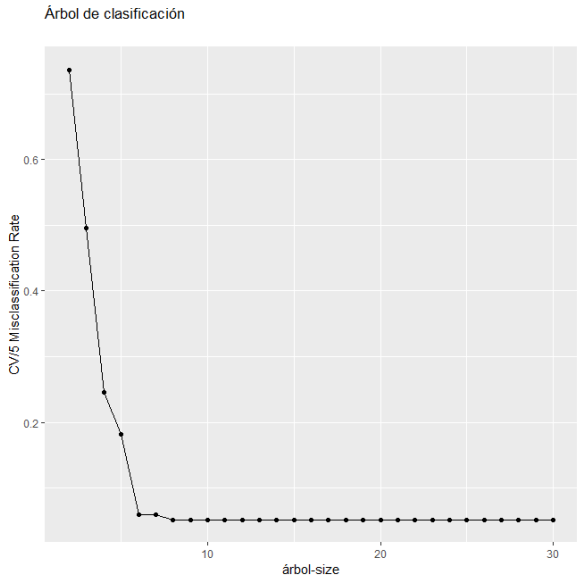

Figure 1.12: Árbol de clasificación K-fold error de clasificación para datos de prueba

Con Random Forest, el error de clasificación para datos de prueba se calcula de la misma manera que en los métodos anteriores, en este caso varía sobre el número de variables utilizadas y el número de iteraciones[7]. En la figura 1.13, se observa que el menor error ocurre al utilizar sólo una variable para el *boostrap* y 200 iteraciones.

[7]Número de árboles a crecer

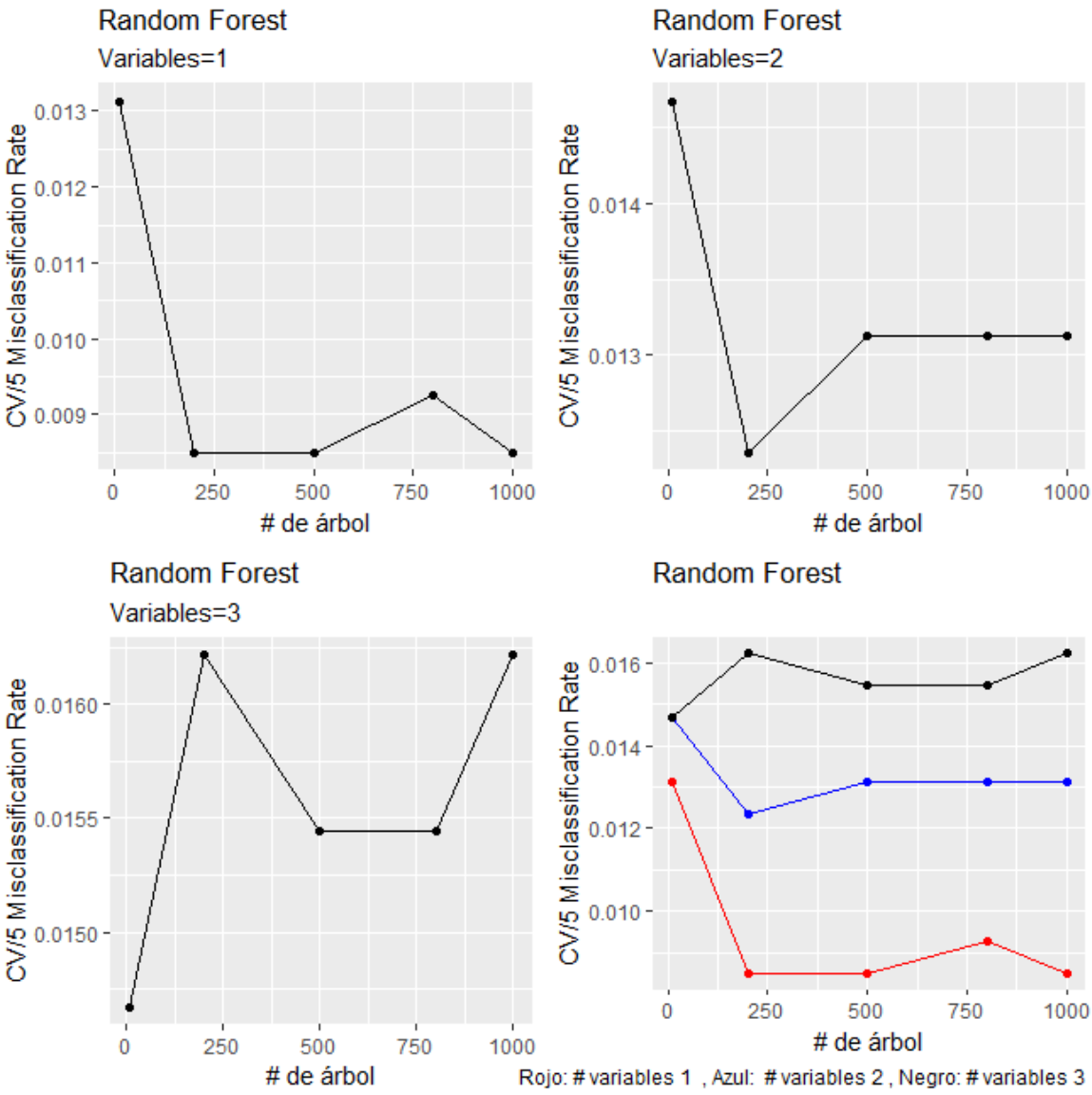

Figure 1.13: Random Forest K-fold error de clasificación para datos de prueba

La figura 1.14, muestra el error de prueba del método Adaboost, utilizando su extensión multiclase SAMME; se observa que el menor error se consigue al implementar 20 clasificadores débiles (o iteraciones).

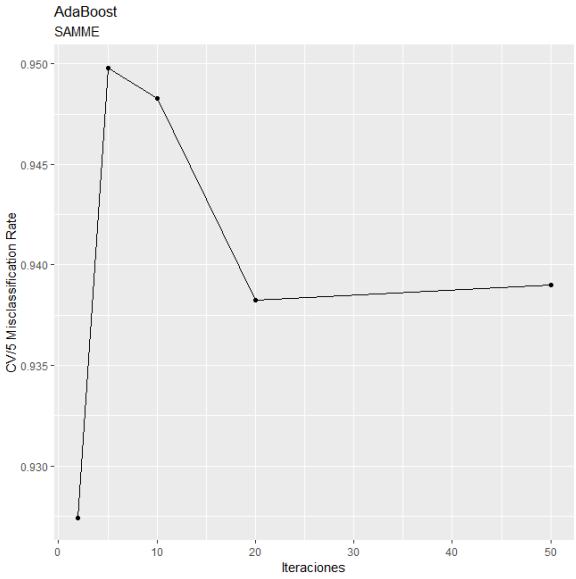

Figure 1.14: AdaBoost K-fold error de clasificación para datos de prueba

Recapitulando, se utiliza SVM con un kernel gaussiano y parámetro constante con valor de siete; el árbol de decisión tiene participación máxima de 6; Random Forest con 200 iteraciones (clasificadores débiles) y una variable para el *boostrap*; AdaBoost con 20 árboles como clasificadores débiles (iteraciones). En la tabla 1.2, se presenta el error de clasificación para los datos de prueba y los modelos previamente mencionados.

	SVM	Árbol clasificador	Random Forest	AdaBoost
Precisión	98.5%	90.2%	98.4%	98%
Error	1.5%	9.8%	1.6%	2%

Table 1.2: K-fold CV/5; Desempeño de los clasificadores con datos de prueba

El método con menor error[8] es SVM con 1.5%, seguido de Random Forest con un error de clasificación de 1.6%, el que registra un mayor error es el árbol de clasificación con un error de prueba de 9.8%.

Para visualizar la clasificación con SVM, en la figura 1.15, se utiliza PCA para reducir dimensiones en los datos de entrenamiento y de prueba, donde las cruces negras son los scores de los datos de prueba proyectados a las dos primeras componentes. En la figura 1.16, se implementa el clasificador SVM previamente mencionado, con el cual se entrena con los scores de los datos de entrenamiento y se prueban con los scores de los datos de prueba, respetando los parámetros previamente seleccionados; se observa que las fronteras de clasificación generaliza a la clasificación de las frutas, presentando un error de prueba es de 0.411.

[8]Se utiliza k-fold cross validation con $k = 5$

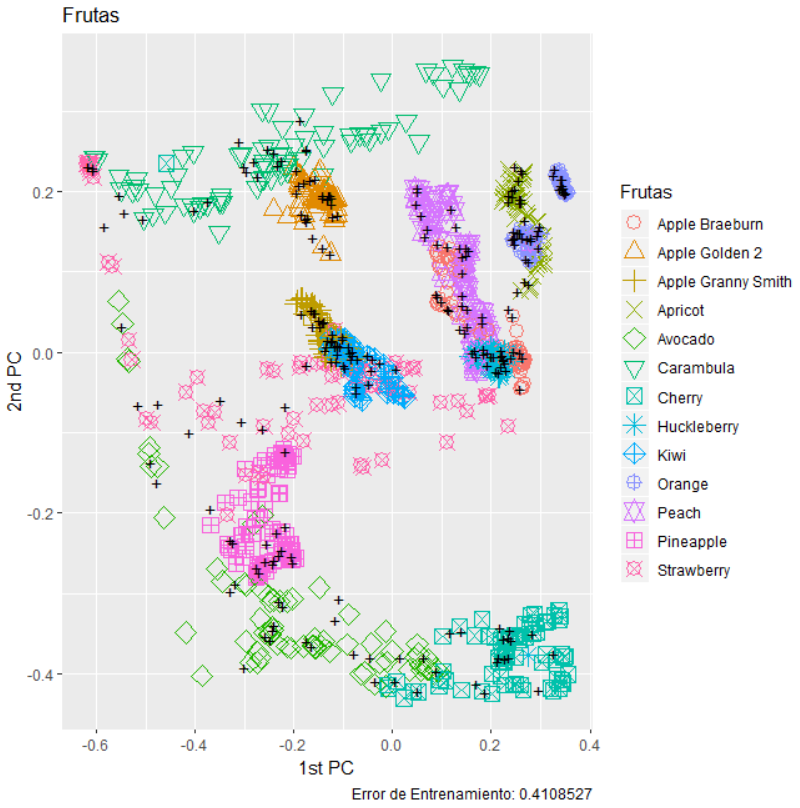

Figure 1.15: Representación en las dos primeras componentes y clasificación con SVM

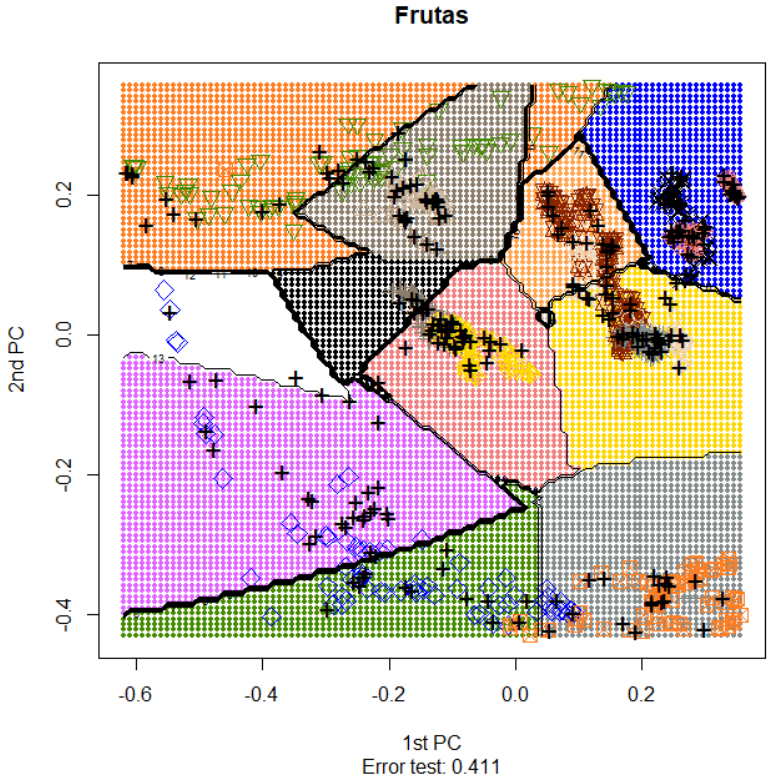

Figure 1.16: Clasificación con SVM; frontera de clasificación

Cuartiles centrales canales HSV[9]

Se utilizan los cuartiles centrales de la mediana de los canales HSV, y en contraste del ejercicio anterior, se cuenta con 9 variables. A diferencia del caso previo, se realiza la clasificación utilizando las primeras dos componentes, esto con fines prácticos y trabajar con dos representaciones distintas.

A los datos de entrenamiento se les aplica PCA, al obtener las cargas se proyectan los datos de prueba a la dimensión de las dos primeras componentes. En la figura 1.17, se observa la representación de las 9 variables en las dos primeras componentes, donde se puede visualizar una clara separación sobre los subconjuntos de frutas.

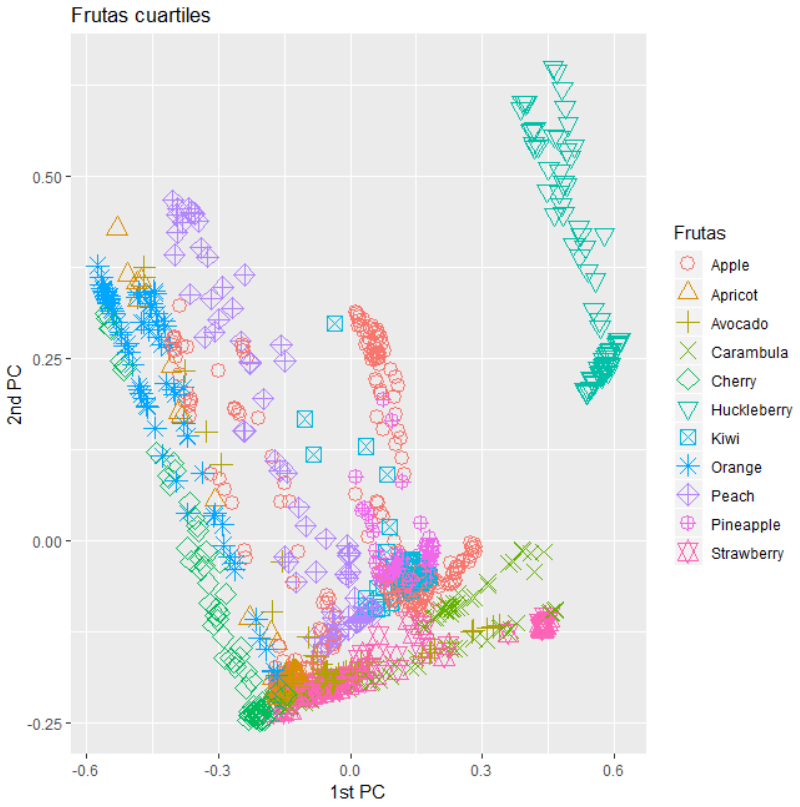

Figure 1.17: Representación en las primeras dos componentes

En la figura 1.18, se aplica SVM y se calcula el error de clasificación por medio de k-fold cross validation, esto para los datos de prueba; a su vez, se utiliza un kernel gaussiano y se varía el valor de su parámetro. El menor error ocurre cuando el parámetro del kernel (gamma), es de 20.

[9]Nota: en caso de utilizar un valor para el parámetro de complejidad será indicado de manera explicita, en caso de no hacerlo el lector debe dar por hecho que no se utilizó alguno.
Nota: no se utilizan la Redes Neuronales dado a problemas en el ajuste del clasificador
Nota: cuando se hizo este ejercicio se juntaron los tipos de manzanas en una clase

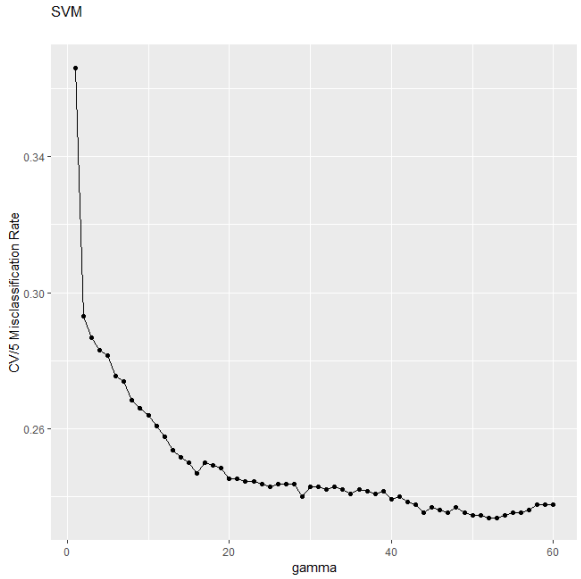

Figure 1.18: SVM K-fold error de clasificación para datos de prueba

En la figura 1.19, se presenta el error de clasificación para el árbol de decisión, variando el tamaño máximo permitido del árbol. Se observa que el menor error de prueba se da con una altura máxima de 8.

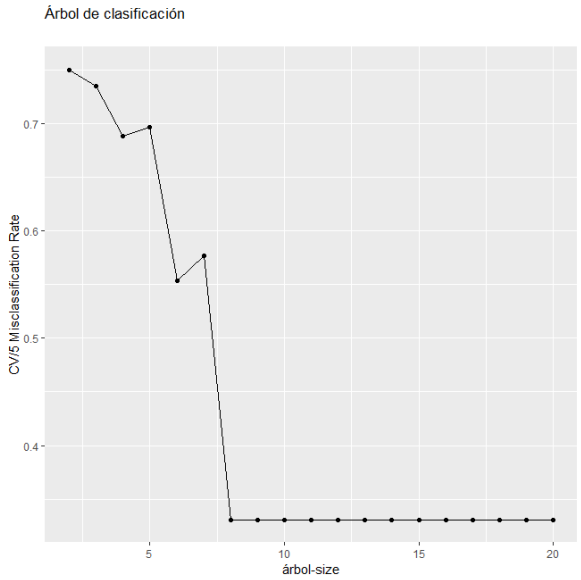

Figure 1.19: Árbol de clasificación K-fold error de clasificación para datos de prueba

Para el método Random Forest, en la figura 1.20, se observa su error de clasificación para el conjunto de datos de prueba. Al variar el número de covariables para el *boostrap* y el número de árboles a utilizar (iteraciones), el error mínimo calculado con k-fold CV, se visualiza al utilizar solo una variable y 200 árboles como máximo.

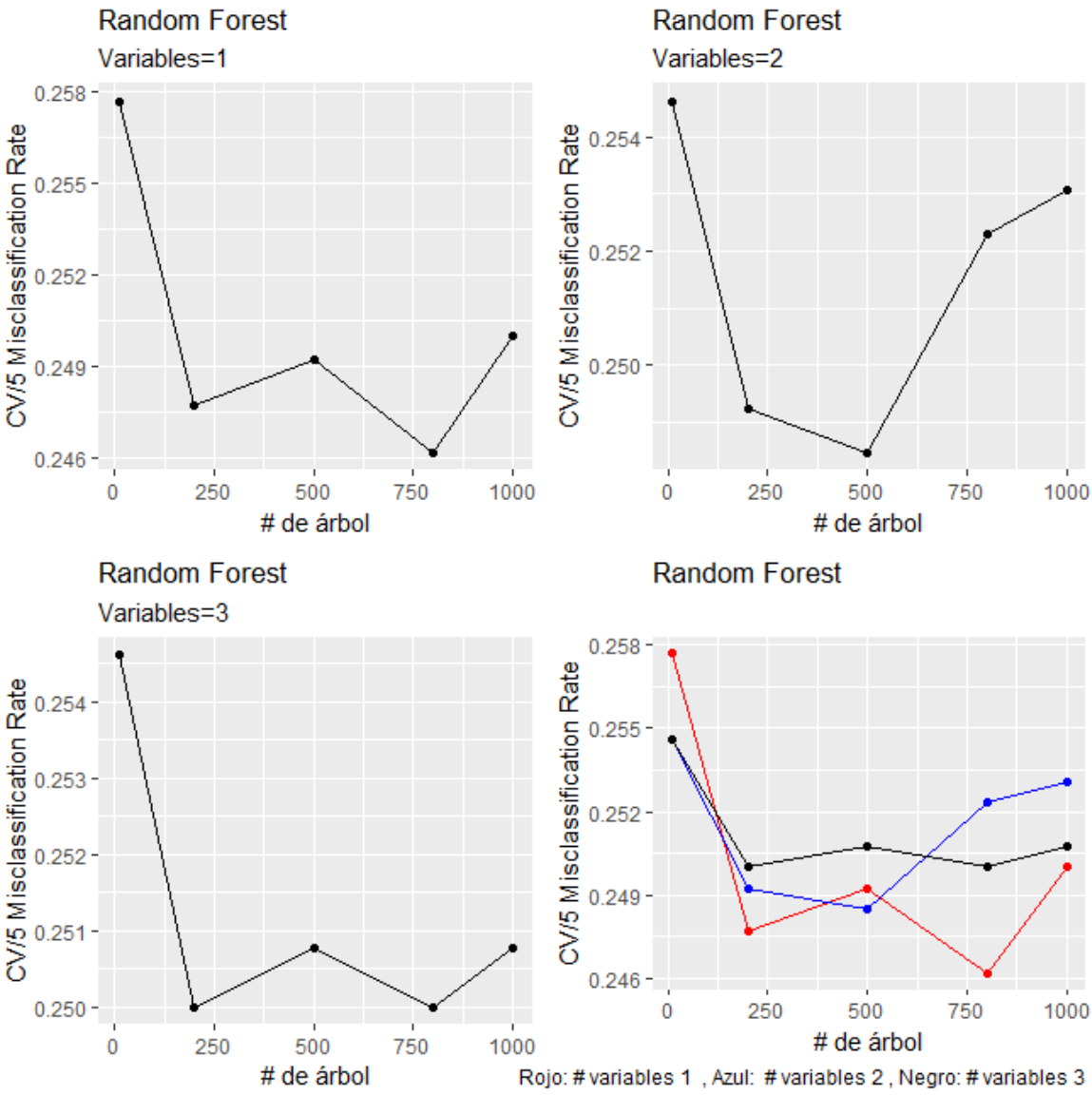

Figure 1.20: Random Forest K-fold error de clasificación para datos de prueba

Por último se implementa adaBoost con la extensión multiclase SAMME, donde su error de clasificación se observa en la figura 1.21, variando el número de clasificadores débiles a utilizar; i.e., número de iteraciones. El error de prueba más pequeño se registra al utilizar 20 iteraciones.

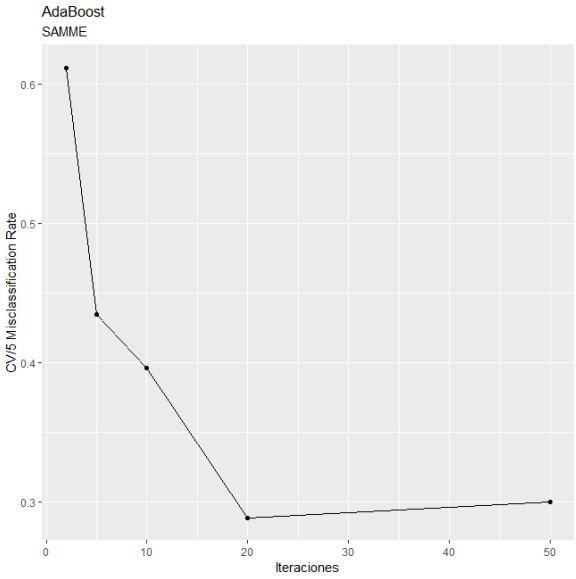

Figure 1.21: AdaBoost K-fold error de clasificación para datos de prueba

Recapitulando, se utiliza SVM con kernel gaussiano y parámetro 20; árbol de clasificación con altura máxima de seis, Random Forest con una variable para el *boostrap* y 200 iteraciones (o número de parboles); AdaBoost con 20 clasificadores débiles.

En la tabla 1.3, se muestra el error de prueba al clasificar medido con k-fold cross validation ($k = 5$),ára los modelos mencionados; el de menor error se presenta con el clasificado SVM con valor de 75.5%, seguido de Random Forest con 74.8%.

En conclusión, utilizar los cuartiles de los canales HSV representados con su media, reporta errores de clasificación más altos para los datos de prueba; sin embargo, esto se puede deberse a que sólo se utilizaron las primeras dos componentes.

	SVM	Árbol clasificador	Random Forest	AdaBoost
Precisión	75.5%	68.8%	74.8%	71%
Error	24.5%	31.2%	25.2%	29%

Table 1.3: K-fold CV/5; Desempeño de los clasificadores con datos de prueba

Para visualizar la clasificación con SVM, en la figura 1.22, se utiliza PCA para reducir dimensiones en los datos de entrenamiento y de prueba, donde las cruces negras son los scores de los datos de prueba proyectados en las dos primeras componentes. En la figura 1.23, se implementa el clasificador SVM previamente mencionado, con el cual se entrena con los scores de los datos de entrenamiento y se prueban con los scores de los datos de prueba, se observa que las fronteras de clasificación tiende a sobre ajustar, provocando posibles errores en la clasificación de los datos de prueba.

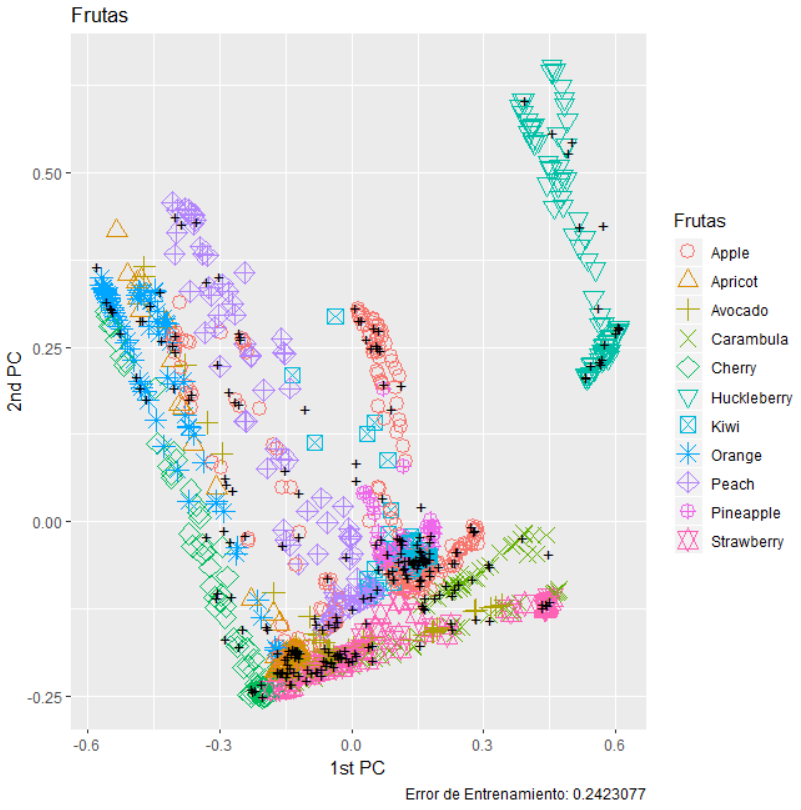

Figure 1.22: Clasificación con SVM

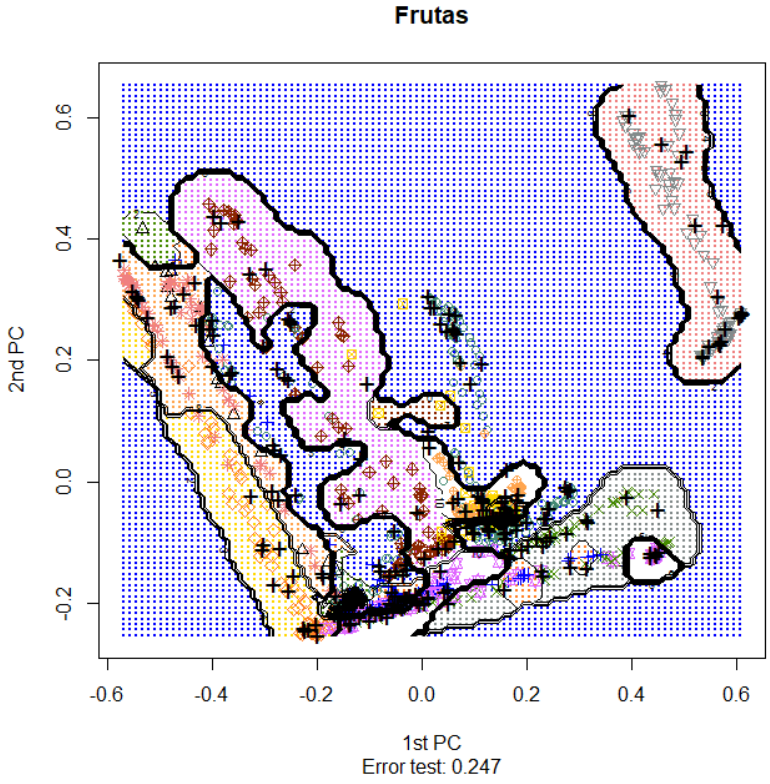

Figure 1.23: Clasificación con SVM; frontera de clasificación

1.4 Solución inciso "d"

Se verifica el desempeño del clasificador que se eligieron en ejemplos reales; no obstante se obtuvieron resultados no muy buenos. Se tomó fotos a una manzana y un aguacate en diferentes posiciones; se utiliza el código en C que la Maestra Karen proporcionó a la clase.

Al igual que en el ejercicio anterior, se utiliza PCA sobre los datos de entrenamiento y se proyectan los de prueba; i.e, las seis imágenes que he tomado. Después se proyectan sobre el espacio de las componentes. En la figura 1.25, se observa la clasificación de las seis imágenes, se puede ver que algunas manzanas las clasifica como starberry, y solo un aguacate se clasifica cercano al grupo de aguacates, las imagen de la esquina extrema con el aguacate lo clasifica con el grupo de manzanas rojas.

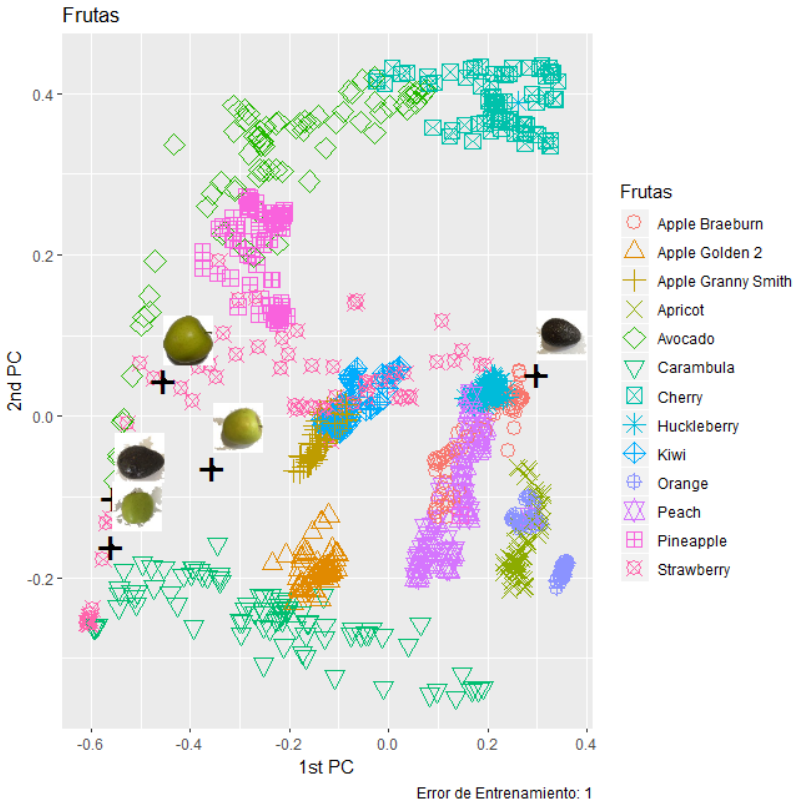

Figure 1.24: Clasificación con SVM; Imagenes reales canales HSV mediana

A raíz de la pobre clasificación, se utilizan los cuartiles de los canales HSV, y de la misma forma que la descrita arriba, se clasifican las seis imágenes bajo esta representación. Se observa que una manzana verde la clasifica cerca a las manzanas y naranjas; las imagenes del extremo derecho del gráfico, se tiene que ya sean manzanas o aguacates el clasificador las agrupa cerca a la carambula.

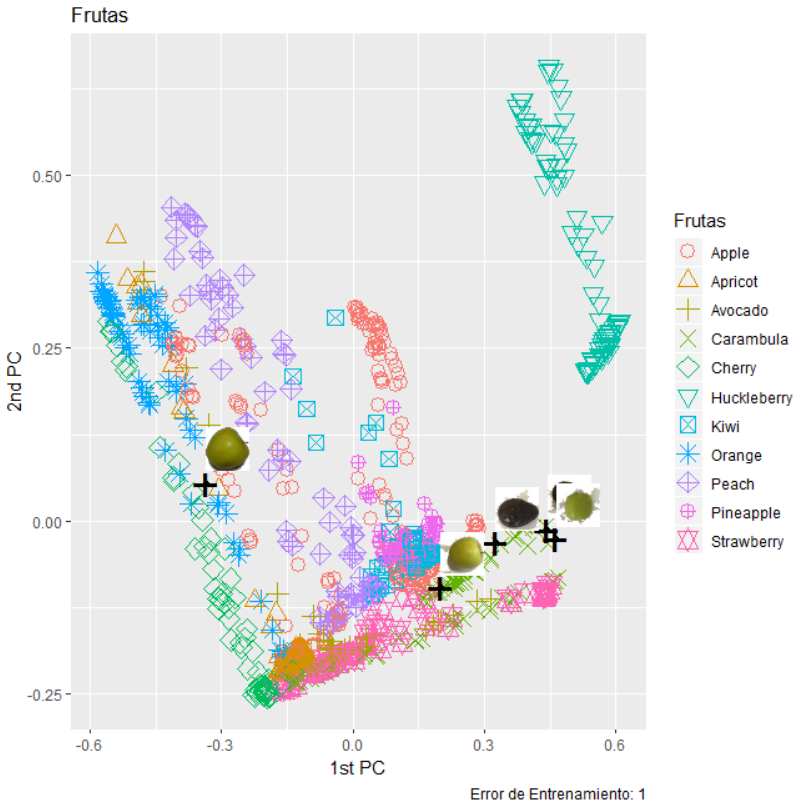

Figure 1.25: Clasificación con SVM; Imagenes reales cuartiles canales HSV mediana

En conclusión los clasificadores elegidos no clasificaron bien las imagenes reales, debido al pobre ajuste o al pre-proceso en la imagen.