Maestría en Electrónica
Reconocimiento de Patrones
Práctica 2: Métodos de Aprendizaje Automático
II Cuatrimestre, 2018

Esteban Martínez Valverde
estemarval@gmail.com

1. Estudio sobre los métodos supervisados y no-supervisados

Como parte de la *Práctica 2* y *Apuntes 2* asignadas, se provee el documento con la investigación de los métodos supervisados y no-supervisados vistos en el curso, a través del siguiente hipervínculo:

https://github.com/Minimal88/Tarea2_RecPatrones/blob/master/docs/Apuntes_2.pdf

2. Selección de un conjunto de datos

Por lo general, la obtención o la selección del conjunto de datos (*Dataset*) siempre es el proceso que requiere de más tiempo en los proyectos de reconocimiento de patrones o *Machine Learning* (ML), ya que la respuesta a los diferentes métodos de ML depende de la naturaleza del *Dataset* escogido. Es por esto que se realizó una extensiva búsqueda en los diferentes repositorios de libre acceso. Los *Datasets* que se identificaron de interés se descargaron, analizaron y realizaron pequeñas pruebas para determinar el comportamiento de los datos.

Finalmente, se encontró un *Dataset* denomidado como *Forest type mapping Data Set*, en el repositorio de ML en linea de la Universidad de California Irvine. Este *Forest Dataset* contiene datos de teledetección (*remote sensing*) multitemporal de un área boscosa en Japón. El objetivo es mapear diferentes tipos de bosques utilizando datos espectrales.

Estos datos fueron obtenidos de un estudio de teledetección que mapeó diferentes tipos de bosque en función de sus características espectrales en longitudes de onda de infrarrojo visible a cercano, utilizando imágenes de satélite ASTER.El resultado (mapa de tipo de bosque) se puede usar para identificar y/o cuantificar los servicios del ecosistema (por ejemplo, almacenamiento de carbono, protección contra la erosión) proporcionados por el bosque.

Los datos se encuentran en dos archivos (training/testing) en el formato *Comma-separated values* (CSV), con las siguientes características:

- Nombre: **"training.csv"**
  - Atributos: 27
  - Instancias: 325 (62 %)
- Nombre: **"testing.csv"**
  - Atributos: 27
  - Instancias: 198 (62 %)

De lo cuál se crea un nuevo archivo (llamado **"forest_dataset.csv"**), conteniendo todos los datos de los dos archivos anteriores, con un total de 523 instancias y 27 atributos Esto, con el fin de poder obtener un mejor control en cuanto a porcentajes entre los *training/testings Datasets*.

La descripción de los atributos se detalla a continuación:

- **Class:**
  Atributo objetivo (Target), para la clasificación en los tipos de bosques encontrados en la zona boscosa de Japón. Los tipos se etiquetan con las siguientes letras:
  - 's'('Sugi'forest)
  - 'h'('Hinoki'forest)
  - 'd'('Mixed deciduous'forest)
  - ó'(Ótherñon-forest land)
- **b1 - b9:**
  Bandas de imagen ASTER que contienen información espectral en las longitudes de onda verde, roja e infrarroja cercana durante las tres fechas: Sept. 26, 2010; Marzo 19, 2011; Mayo 08, 2011.
- **pred_minus_obs_S_b1 - pred_minus_obs_S_b9:**
  Valores espectrales obtenidos de la predicción (basados en interpolación espacial) menos valores espectrales reales para la clase de bosque tipo 's'(b1-b9).
- **pred_minus_obs_H_b1 - pred_minus_obs_H_b9:**
  Valores espectrales obtenidos de la predicción (basados en interpolación espacial) menos valores espectrales reales para la clase de bosque tipo 'h'(b1-b9).

Se realiza una codificación de las clases para tener valores numericos y no texto, ya que se presentaron problemas con algunos de los metodos. A continuación se muestra el cambio de valores realizado:

- 0 : 's'('Sugi'forest)
- 1 : 'h'('Hinoki'forest)
- 2 : 'd'('Mixed deciduous'forest)
- 3 : ó'(Ótherñon-forest land)

3. Aplicación de métodos de aprendizaje

Se realizaron implementaciones de siete métodos de aprendizaje para ML supervisado. Para ello se utilizó el paquete de *python Sklearn*. El código fuente de los métodos de aprendizaje se pueden encontrar en la carpeta llamada **Source** en formato de un *Jupyter Notebook* (.ipynb), el acceso es a través del siguiente repositorio de *GitHub*:

https://github.com/Minimal88/Tarea2_RecPatrones

La lista de los archivos viene enumerada de acuerdo a cada subsección que se presenta a continuación, con el objetivo de explicar los resultados obtenidos de cada implementación.

Para todas las pruebas realizadas se utilizó 80 % de las instancias para el entrenamiento y el 20 % para las pruebas. Cabe destacar que debido a la poca cantidad de instancias, es posible que los resultados varíen (con respecto a los presentados en este documento) cuando se vuelva a ejecutar el código implementado.

3.1. kNN *classification*

El *Jupyter notebook* de este método se puede encontrar con el nombre: **1.kNN_classification_sklearn[Forest].ipynb**

En la Figura 1, se muestra la presunción en el proceso de entrenamiento y pruebas (*training/testing*) en función de la cantidad de vecinos (k). Como se observa el mejor resultado intermedio entre entrenamiento y pruebas es cuando se obtiene un (k=3).

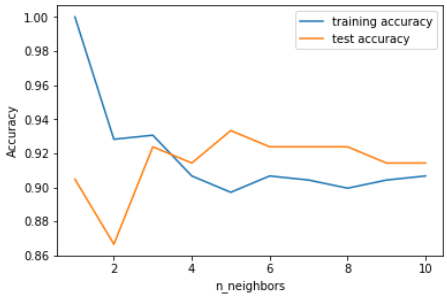


Figura 1: Precisión de Training/testing en función de la cantidad de vecinos (k), para el método kNN

Por lo que se procede obtener una curva de aprendizaje (*learning curve*), a través de una validación cruzada (CV,*Cross-Validation*) de los datos con K=3. El resultado se muestra en la Figura 2, y se observa que a partir de las **100 muestras** el CV, obtiene una precisión mayor de **0.85**.

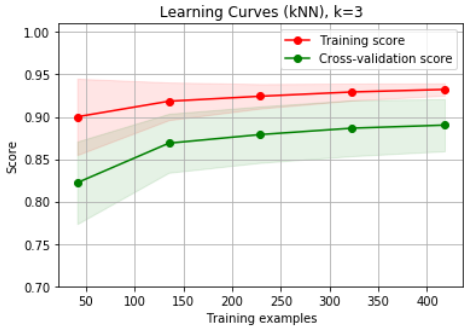


Figura 2: Curva de aprendizaje obtenido del *Cross-Validation* para el método kNN, con k=3.

Además se realiza una inspección visual de las lineas de decisión generadas con k=3, utilizando las 4 clases disponibles en el *Dataset*. En este caso se realizado de k=[1-4], para ver el comportamiento de los primer 4 casos. El resultado se muestra en la Figura 3, y se puede observar como en k=4 las lineas de decisión pierden la precisión.

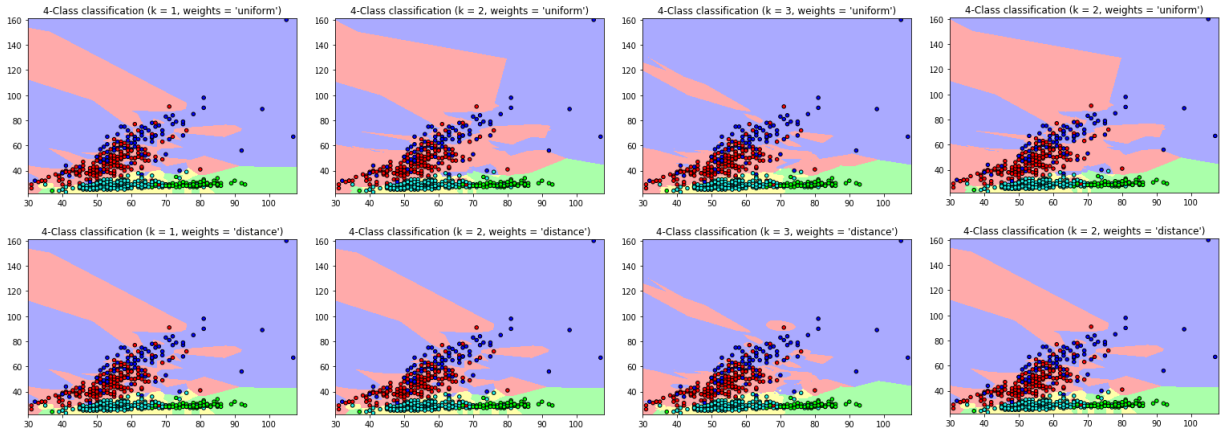
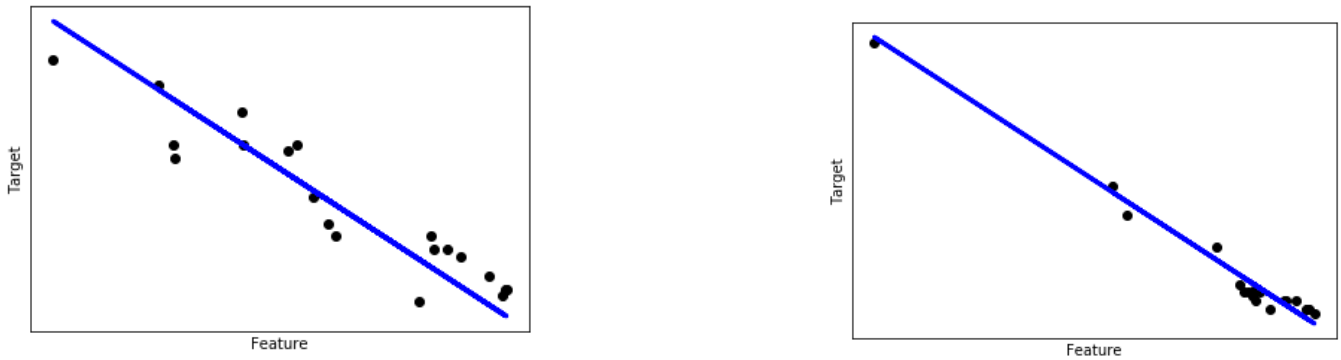


Figura 3: Visualización de las barreras de desición para la clasificación kNN, para K=[1-4].

3.2. *Linear Regression*

El *Jupyter notebook* de este método se puede encontrar con el nombre: **2.linear_regression_sklearn[forest].ipynb**.

La característica de linealidad de los datos se desconoce, por lo que se crea un código que permita obtener las diferentes precisiones obtenidas tras realizar la predicción entre una columna y el resto de columnas dentro del *Dataset*. De esta manera es posible obtener la linealidad no solo entre el tipo de bosque (el atributo Class) y los demás atributos , sino que también entre todos los atributos. Esto se realizó de esta manera ya que preliminarmente, se identificó que la relación entre el atributo Class y los demás atributos no presentaban linealidad. Se realizaron pruebas con los atributos 'b1'y 'b2'. Se obtuvo un precisión entre 'b1'y 'b9'de **0.82**, mientras que la precisión entre 'b2'y 'pred_minus_obs_H_b2', fue de **0.97**. El resultado de la predicción en el caso de 'b1'se presenta en la Figura 4a y en el caso de 'b2', se presenta en la Figura 4b.

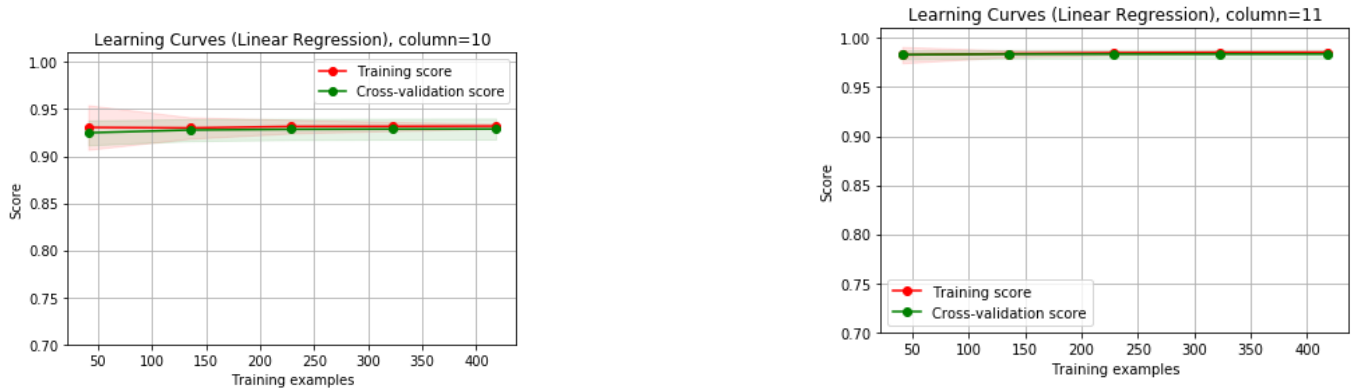


(a) Atributos 'b1'y 'b9'

(b) Atributos 'b2'y 'pred_minus_obs_H_b2'

Figura 4: Predicciones con regresión lineal entre los diferentes atributos.

Se procede a obtener las curvas de aprendizaje con CV para ambos modelos de regresión lineal. En la Figura 5a se presenta el caso para 'b1', donde se observa que se obtiene una precisión mas de **0.9**, desde las **50** muestras. Y en el caso para 'b2'se observa una presición mayor de **0.95** igual desde las **50** muestras. Siendo congruentes estos resultados con las precisiones obtenidas sin CV.



(a) Atributos 'b1'y 'b9'.

(b) Atributos 'b2'y 'pred_minus_obs_H_b2'.

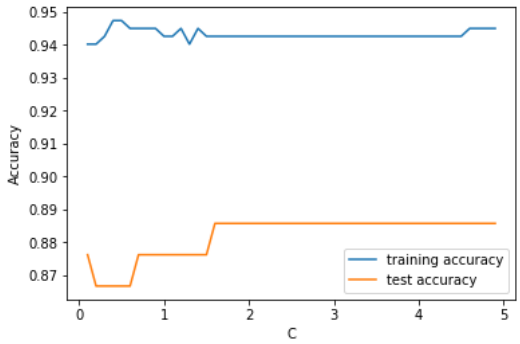
Figura 5: Curvas de aprendizaje obtenidas del *Cross-Validation* para la regresión lineal en los atributos 'b1'y 'b2'.

Se puede decir que el comportamiento de la predicción lineal en los procesos de entrenamiento y pruebas entre las variables seleccionadas es bastante buena (más de 0.9). Por lo que estas predicciones se pueden usar indirectamente para determinar la clasificación del tipo de bosque. Como no se tiene un comportamiento directo

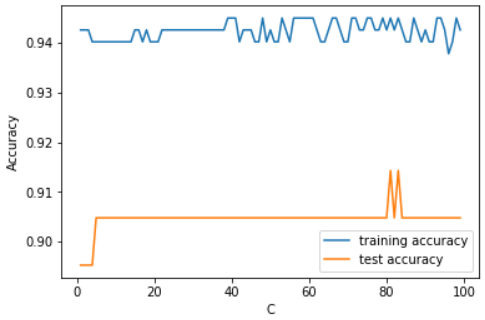
3.3. *Logistic Regression*

El *Jupyter notebook* de este método se puede encontrar con el nombre: **3.logistic_regression_sklearn[forest].ipynb**

En el método de la regresión logística se enfocó en encontrar el parámetro 'C', que equivale al inverso de la fuerza de regularización. Un valor alto de 'Címplica menos regularización, el modelo trata de ajustarse a los datos de training lo mayor posible. Mientras que un valor bajo de 'Címplica encontrar coeficientes w cercanos a cero. Por lo que se realizan dos pruebas con dos rangos de valores para 'C': a) De 0.1 a 5 cada 0.1; b) de 1 a 100 cada 1. Los resultados para el rango a) se presenta en la Figura 6a, donde se obtiene una precisión de **0.88** a partir de C=2. Para el rango b), en la Figura 6b, se observa que se obitiene una precisión mayor a **0.91** con un valor en C=83.



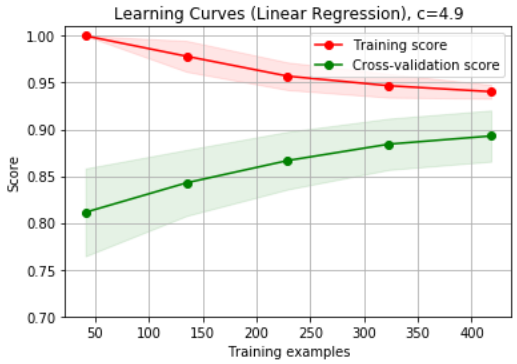
(a) Rango de 0.1 a 5 cada 0.1.



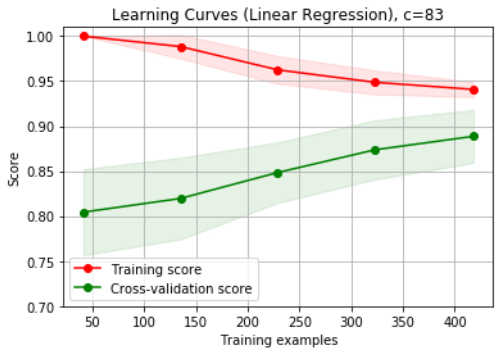
(b) Rango de 1 a 100 cada 1.

Figura 6: Precisión de *Training/Testing* en función del parámetro 'C', con distintos rangos, para el método de regresión logística.

De acuerdo con los valores obtenidos del parámetro 'C', donde se tiene un valor máximo de la precisión en las pruebas, se procede a obtener una comparación del comportamiento en el aprendizaje con CV. En la Figura 7a, se muestra el resultado para el rango a), donde se obtiene una precisión de **0.85** a partir de las **200** muestras. Mientras que para el rango b), en la Figura 7b, se obtiene una precisión de **0.85** a partir de las **300** muestras.



(a) C=4.9



(b) C=83.

Figura 7: Curvas de aprendizaje obtenidas del *Cross-Validation* para la regresión logística para distintos valores de 'C'.

Entonces, se puede decir que el mejor valor encontrado para C de acuerdo las curvas de aprendizaje es C=4.9, ya que a menor cantidad de muestras, presenta la misma precisión que con un C=83.

3.4. Naive Bayes

El *Jupyter notebook* de este método se puede encontrar con el nombre: **4.naive_ayesb_sklearn[forest].ipynb**

Se implementa el método de Naive Bayes, utilizando una distribución gausiana de los datos (GaussianNB). Está función tiene como parámetro de entrada prioridad de probabilidad para cada una de las clases. Si no se especifica el mismo algoritmo ajusta estos parámetros. Por lo que se decide realizar con el ajuste automático y se obtiene una precisión de **0.876** en el **training**, y **0.867** en el **testing**.

Seguidamente, se realiza el *cross-validation* con el modelo de Naive Bayes obtenido. En la Figura 8, se observan las curvas de aprendizaje, y se observa que para el CV se tiene una presición de **0.85** a partir de las **150** muestras.

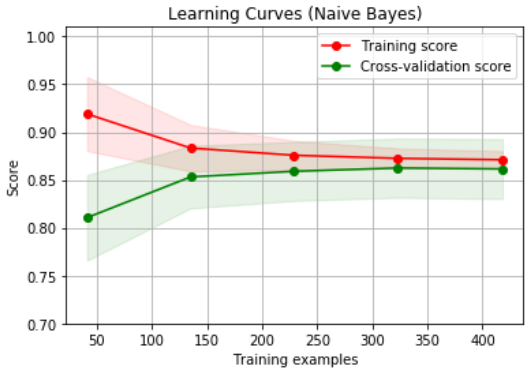
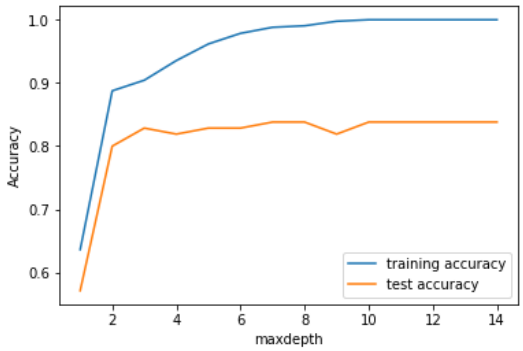


Figura 8: Curva de aprendizaje obtenida del *Cross-Validation* para el método Naive Bayes.

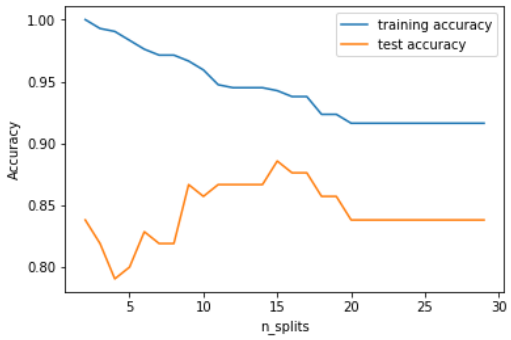
3.5. *Decision Trees*

El *Jupyter notebook* de este método se puede encontrar con el nombre: **5.desicion_tree_sklearn[forest].ipynb**

La implementación del método *Decision Trees*, se realiza considerando obtención de los parámetros: máxima profundidad (*max_depth*) y mínimo de divisiones por muestras (*min_samples_split*). De los cuales se obtienen los resultados del *training/testing* para distintos rangos de los parámetros seleccionados, como se presentan en la Figura 9a. De donde se determinan los mejores valores de acuerdo a los resultados de *testing*.



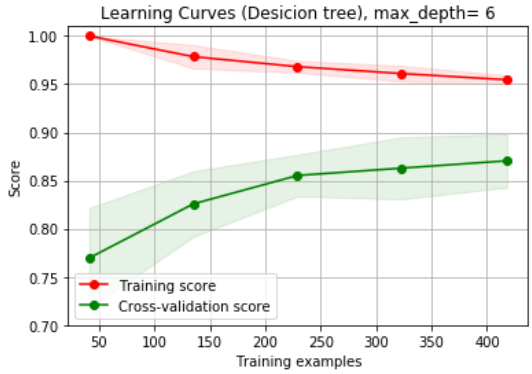
(a) *max_depth* = [1-15]



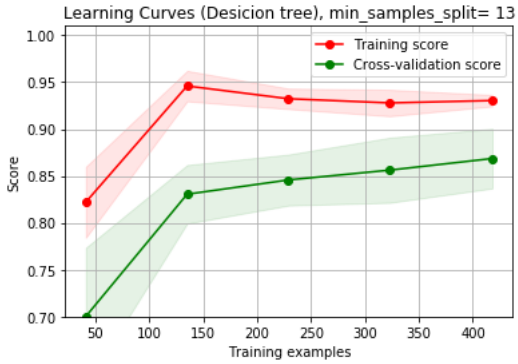
(b) *min_samples_split* = [1-30]

Figura 9: Precisión de *Training/Testing* en función de rangos de los parámetros *max_depth* y min_*samples_split*, para el método de *Decision Trees*.

Se obtienen y muestran las curvas de aprendizaje a través del *cross-validation* para el método de *Decision Trees*, utilizando como parámetros: *max_depth*=6 (Figura 10a)y *min_samples_split*=13 (Figura 10a). Donde se observa que los dos escenarios, con diferentes parámetros, obtienen una precisión de **0.85** después de **225** muestras en el CV.



(a) *max_depth*=6 (Figura 10a)



(b) *min_samples_split*=13.

Figura 10: Curvas de aprendizaje obtenidas del *Cross-Validation* para *Decision Trees*, con distintos valores de *max_depth* y min_*samples_split*.

3.6. *Random Forest*

El *Jupyter notebook* de este método se puede encontrar con el nombre: **6.random_forest_sklearn[forest].ipynb**

Se realiza la implementación del método *Random Forest*, considerando la cantidad de estimadores 'C'. Se utiliza un rango de 1-100, se obtienen los resultados del *training/testing* en la Figura 11. De donde se determinan los mejores valores de acuerdo a los resultados de *testing*.

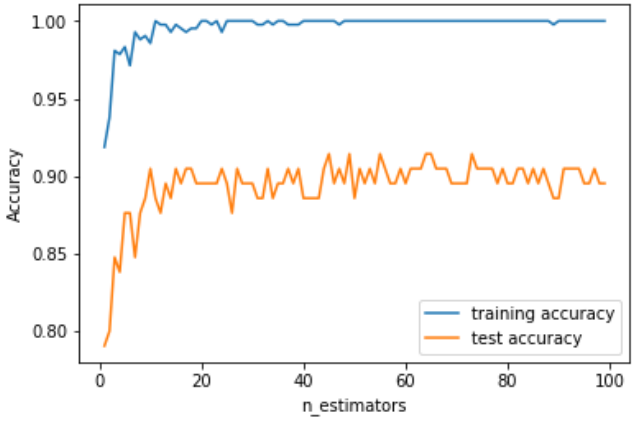
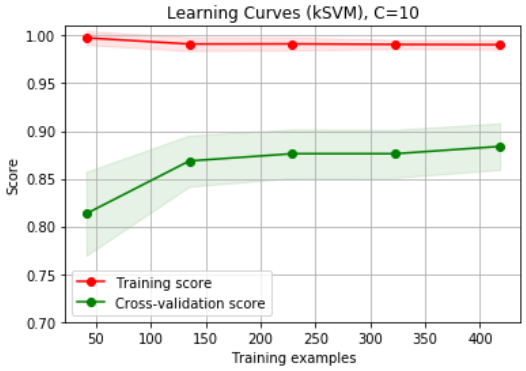
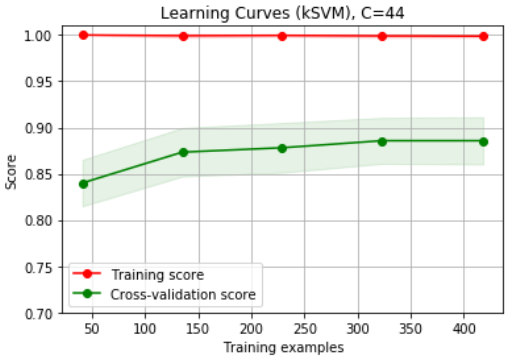


Figura 11: Visualización de las barreras de decisión

Se obtienen y muestran las curvas de aprendizaje a través del *cross-validation* para el método de *Random Forest*, utilizando los valores para la cantidad de estimadores: C=10 y C=44, correspondiendo al mayor puntaje obtenido en el entrenamiento y pruebas respectivamente. En la Figura 12), se observa como se obtiene una precisión de mayor de **0.85** a partir de las **100** muestras en el CV.



(a) n_estimators=44, training

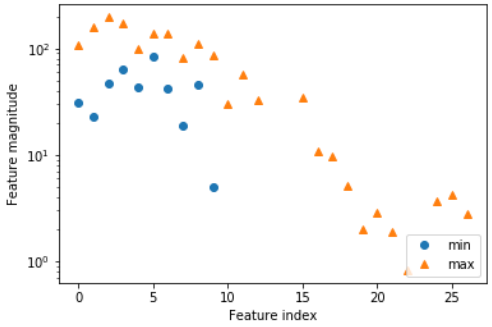


(b) n_estimators=44, testing

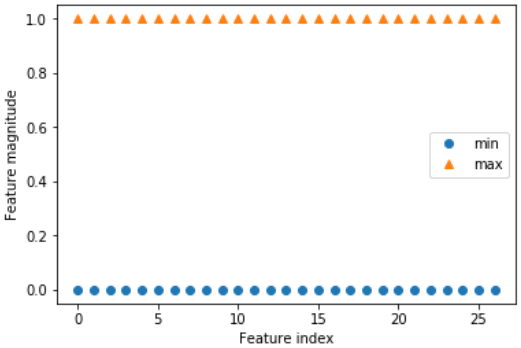
Figura 12: Precisión de *Training/Testing* en función de los estimadores en un rago de 1-100, para el método de *Random Forest*.

3.7. *kernel SVM* (kSVM)

El *Jupyter notebook* de este método se puede encontrar con el nombre: **7.kernel_svm_sklearn[forest].ipynb** Se realiza el proceso de *training/testing* con el método *kernel SVM* utilizando los valores de 'C'y 'gamma', predefinidos. Y se obtiene una precisión para *training*=1.00 y para *testing*=0.38, por lo que se realiza una inspección de los máximos y mínimos (Figura 13a). Luego se hace realiza un escalamiento de los datos en el rango de [0-1] (Figura 13b). Se vuelve a realizar eñ proceso de *training/testing* pero esta vez se obtiene una precisión para *training*=**0.801** y para *testing*=**0.857**.



(a) *dataset* original.



(b) *dataset* con escalamiento.

Figura 13: Visualización de los mínimos y máximos de los *dataset* original y con escalamiento.

Se obtienen las curvas de aprendizaje a través del *cross-validation* para el método de *Kernel SVM*, se muestran en la Figura 14) y se observa como se obtienen una precisión de mayor de **0.85** a partir de las **100** muestras en el CV.

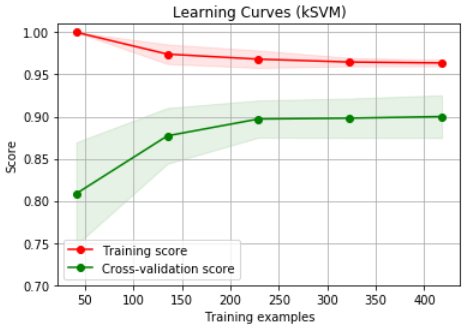


Figura 14: Visualización de las barreras de desición

3.8. Curvas de aprendizaje

Como se muestra en las secciones anteriores, se obtuvieron las curvas de aprendizaje para cada uno de los métodos de ML supervisado, considerando los mejores parámetros encontrados. A manera de resumir los resultados obtenidos con la validación del *cross-validation*, se presentan la precisión asociada a la cantidad de muestras obtenidas para cada uno de los métodos elegidos (Cuadro 1). De lo cuál se puede determinar que los métodos más rapidos en aprender corresponden a: *kNN*, *Linear Regression*, *Random Forest* y *kVSM*. También se puede decir que de todos los métodos implementados, con los parámetros elegidos, es muy poco probable que se obtenga una precisión mayor o igual a 0.9.

Cuadro 1: precisión asociada a la cantidad de muestras obtenidas para cada uno de los métodos elegidos

ML method	**Score**	**Samples**
kNN	0.85	100
Linear Regression	0.9	100
Logistic Regression	0.85	200
Naive Bayes	0.85	150
Desicion Trees	0.85	225
Random Forest	0.85	100
kSVM	0.85	100