

Práctica Azure- Clasificador Películas-

Jorge Casan Vázquez

1. Informe Ejecutivo

El presente informe tiene por objetivo clasificar a las películas como buenas o malas en función de su Rating. Los dos modelos los cuales trabajaremos son Redes Neuronales y Árboles de decisión Boosting. El primer modelo ofrece una precisión del 98,9% con un área por debajo de la curva (AUC) de la curva ROC del 0.999. Por otra parte, con el modelo Boosting nos da una precisión perfecta del 100% con un AUC del 1.00, en consecuencia, con una matriz de confusión en la cual la cantidad de falsos positivos y falsos negativos resulta nula, estando concentrada las observaciones predichas en la parte de verdaderos positivos y falsos negativos. Seguidamente adjuntaremos los códigos en R de ambos modelos generados a través del servicio web.

2. Desarrollo de la práctica

En primer lugar, los datasets con los que trabajaremos son: 'Movie Ratings' y 'Movie Tweets'. Tendremos que juntar ambos datasets en función de la columna común 'MovieId' a ellas con un inner join.

En segundo lugar, considero que Rating (2), para la posterior clasificación de películas, puede resultar ambigua, por lo que prefiero cambiar el nombre de dicha variable por la de Rating2, la cual será la valoración de las películas extraída del dataset 'Movie Tweets'.

En tercer lugar, realizamos nuestra primera Transformación SQL, la cual quiero que me devuelva el identificador de las películas, la media de la valoración de las películas del dataset 'Movie Ratings' y la del dataset 'Movie Tweets', en donde le pongo la condición de los ratings que sean inferiores a 10. De esta forma acotamos los outliers que tenemos del dataset, para finalmente agruparlos por el identificador de las películas.

Cuestión digna de mención es el tratamiento de los outliers, los cuales tienen una influencia muy considerada en la media, pero no en la mediana. Aunque no existe como tal una query que establezca la mediana de los Ratings de ambos datasets, he optado por establecer una condición, la cual acote la valoración de estas hasta un máximo de 10, puesto que la valoración debe de estar comprendida dentro de una escala métrica convencional.

En cuarto lugar, renombro las columnas Rating1 y Rating2 como 'Rating_Movies' y 'Rating_Tweets'.

En quinto lugar, aplico otra transformación SQL, la cual quiero que me devuelva el identificador de la película y las columnas anteriormente renombradas del cuarto paso en donde para aquellas películas cuyo rating sea superior a 7 sea considerada como buena y en caso contrario tendrá una mala valoración. De la escala métrica, he optado clasificar a las películas como buenas o malas en función de si tienen un Rating superior a un 7 porque un notable en su valoración es condición suficiente, toda esta query la nombro como 'Etiqueta'

En sexto lugar, aplicamos el Split Data, en donde particionamos los datos muestrales los cuales contienen el 80% del total de observaciones y el resto el test.

En séptimo lugar, con el modelo de entrenamiento filtro por ‘Etiqueta’, la cual ha sido calculada en el paso quinto, y a partir de este momento calculo el ‘score model’, en base a los modelos ‘Two-Class Neuronal Network’ y ‘Two-Class Boosted Decision’.

Por último, evalúo ambos modelos y obtengo los resultados.

Con la red neuronal:

True Positive	False Negative	Accuracy	Precision
1495	23	0.985	0.989
False Positive	True Negative	Recall	F1 Score
17	1188	0.985	0.987
Positive Label	Negative Label		
Mala	Buena		

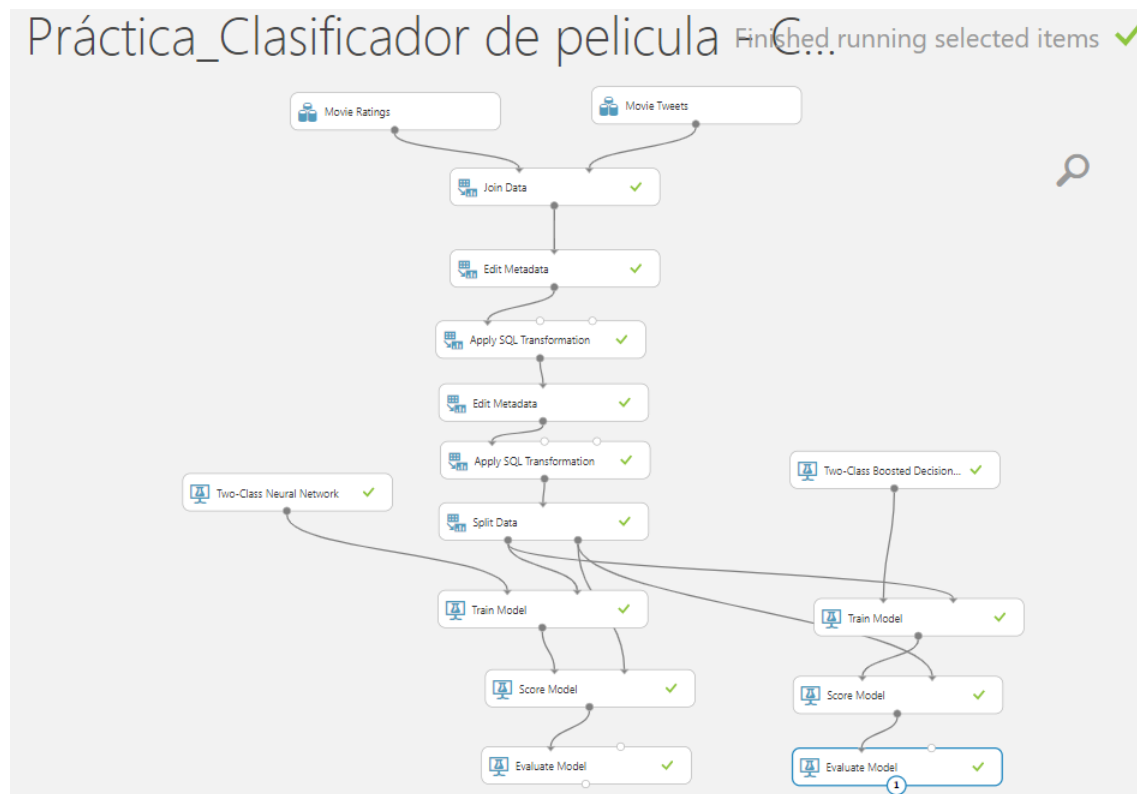
Con el modelo de la red neuronal obtenemos una precisión del 98,9%, un AUC del 0.999. En cuanto a la matriz de confusión obtengo 17 películas las cuales se clasifican como falsos positivos y 23 como falsos negativos.

Con el modelo Boosting:

True Positive	False Negative	Accuracy	Precision
1518	0	1.000	1.000
False Positive	True Negative	Recall	F1 Score
0	1205	1.000	1.000
Positive Label	Negative Label		
Mala	Buena		

Con el modelo boosting obtenemos una precisión perfecta del 100%, con un AUC perfecto, en donde la matriz de confusión recoge el total de las observaciones predichas, las cuales las clasifica solamente como verdaderos positivos y verdaderos negativos.

Adjunto podemos ver todos los pasos puestos en producción en Azure, para la consecución de dicho objetivo:

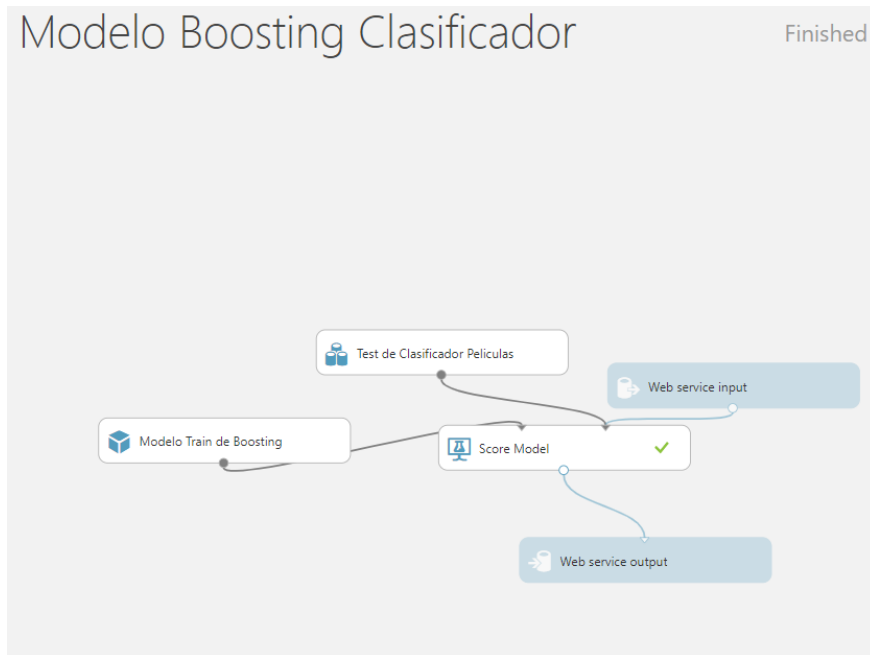


<https://studio.azureml.net/Home/ViewWorkspaceCached/a7918aea5ed44a64b027be4e19c822c9#Workspaces/Experiments/Experiment/a7918aea5ed44a64b027be4e19c822c9.f-id.0063ab1a9f134527b6a4b4a8a271a61a/ViewExperiment>

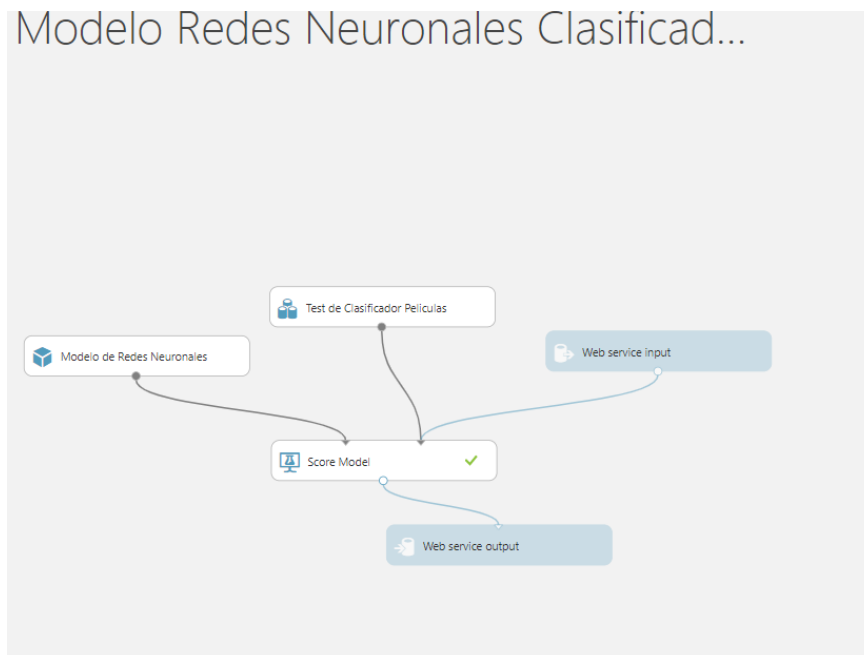
Ahora vamos a ver cómo generar el servicio web de nuestros modelos.

Para ello, nos crearemos dos nuevos experimentos, para ambos proyectos. El test será igual para ambos, sin embargo, el modelo escogido dependerá de qué modelo se haya optado para realizar la clasificación.

A continuación, se puede ver como se ha hecho:



Por otra parte, se ha hecho lo mismo para la red neuronal:



3. Conclusiones

Esta práctica ha sido de gran ayuda a la hora de establecer un criterio de clasificación según los identificadores de las películas. Además de evaluar cada uno de los modelos he podido comparar entre sus diferentes matrices de confusión, sus AUC, su curva ROC, su precisión y su recall, para finalmente decantarme por el modelo boosting, ya que establece un criterio de clasificación perfecto, si bien, el modelo diseñado por la red neuronal es muy bueno.

4. Anexos Códigos R para ambos modelos

He decidido anexar el código R, aunque el servicio web también genera el modelo en formato C y en Python.

El código generado en R, para nuestro clasificador de películas según la red neuronal es el siguiente:

```
library("RCurl")
library("rjson")

# Accept SSL certificates issued by public Certificate Authorities
options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))

h = basicTextGatherer()
hdr = basicHeaderGatherer()

req = list(
  Inputs = list(
    "input1"= list(
      list(
        'MovieId' = "1",
        'Rating_Movies' = "1",
        'Rating_Tweets' = "1",
        'Etiqueta' = ""
      )
    )
  ),
  GlobalParameters = setNames(fromJSON('{}'), character(0))
)
```

```

body = enc2utf8(toJSON(req))

api_key = "abc123" # Replace this with the API key for the web service
authz_hdr = paste('Bearer', api_key, sep=' ')

h$reset()

curlPerform(url = "https://ussouthcentral.services.azureml.net/workspaces/a79
18aea5ed44a64b027be4e19c822c9/services/0d18211bc13e44439ec7128656de0316/execu
te?api-version=2.0&format=swagger",
httpheader=c('Content-Type' = "application/json", 'Authorization' = authz_hdr
),
postfields=body,
writefunction = h$update,
headerfunction = hdr$update,
verbose = TRUE
)


headers = hdr$value()
httpStatus = headers["status"]
if (httpStatus >= 400)
{
print(paste("The request failed with status code:", httpStatus, sep=" "))

# Print the headers - they include the request ID and the timestamp, which ar
e useful for debugging the failure
print(headers)
}

print("Result:")
result = h$value()

print(fromJSON(result))

```



Por otra parte, el modelo generado en R para el boosting es el siguiente:

```

library("RCurl")
library("rjson")

```

```

# Accept SSL certificates issued by public Certificate Authorities

options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))

h = basicTextGatherer()
hdr = basicHeaderGatherer()

req = list(
  Inputs = list(
    "input1"= list(
      list(
        'MovieId' = "1",
        'Rating_Movies' = "1",
        'Rating_Tweets' = "1",
        'Etiqueta' = ""
      )
    )
  ),
  GlobalParameters = setNames(fromJSON('{}'), character(0))
)

body = enc2utf8(toJSON(req))

api_key = "abc123" # Replace this with the API key for the web service
authz_hdr = paste('Bearer', api_key, sep=' ')

h$reset()

curlPerform(url = "https://ussouthcentral.services.azureml.net/workspaces/a7918aea5ed44a64b027be4e19c822c9/services/fe186365a9b74f5996965bc81e6456a5/execute?api-version=2.0&format=swagger",
  httpheader=c('Content-Type' = "application/json", 'Authorization' = authz_hdr),
  postfields=body,
  writefunction = h$update,
  headerfunction = hdr$update,
  verbose = TRUE

```

```

)

headers = hdr$value()
httpStatus = headers["status"]
if (httpStatus >= 400)
{
print(paste("The request failed with status code:", httpStatus, sep=" "))

# Print the headers - they include the request ID and the timestamp, which are
# useful for debugging the failure
print(headers)
}

print("Result:")
result = h$value()

print(fromJSON(result))

```

Microsoft

Estos son los links:

<https://services.azureml.net/workspaces/a7918aea5ed44a64b027be4e19c822c9/webservices/ad39d0a6874b44279dab58915110368d/endpoints/default/consume>

<https://services.azureml.net/workspaces/a7918aea5ed44a64b027be4e19c822c9/webservices/f40cb3598f02480297a99ee6b9b31966/endpoints/default/consume>

5. **Bibliografía de apoyo**

La siguiente bibliografía la he utilizado a modo de aprendizaje personal y a modo didáctico:

<https://azure.microsoft.com/es-es/services/machine-learning-studio/>

<https://docs.microsoft.com/es-es/azure/machine-learning/>

<https://github.com/Azure/MachineLearningNotebooks>

<https://www.softeng.es/es-es/blog/los-beneficios-del-aprendizaje-automatico-con-microsoft-azure-machine-learning.html>