



Facultad
de
Informática



UNIVERSIDAD
COMPLUTENSE
MADRID

Microbios
&
Aprendizaje Automático

Álvaro Álvarez Iglesias
David Stetco

Presentación del dataset	3
Preprocesamiento	6
Dataset final	7
Subconjuntos de entrenamiento, validación y test	7
Regresión logística	8
Adaptación	8
Ajuste	8
Resultados	9
Redes neuronales	9
Adaptación	9
Ajuste	10
Resultados	11
Supported vector machine(SVM)	12
Adaptación	12
Ajuste	12
Resultados	13
Comparación y Conclusiones	14
Bibliografía	15

Presentación del dataset

Las nuevas tecnologías de secuenciación del ADN han proliferado en las dos últimas décadas. Las continuas mejoras en la "secuenciación de próxima generación" (NGS) y en la "secuenciación de tercera generación" (TGS) han aumentado la fidelidad y la velocidad de esta, pero todavía se tardan horas o días en obtener secuencias completas. Además, hay algunas aplicaciones diagnósticas en las que se hace imprescindible la identificación muy rápida de un gen o especie genética concreta.

Por ejemplo, en pacientes con shock séptico por infecciones bacterianas, la identificación de los genes de resistencia a los antibióticos es esencial porque la tasa de mortalidad aumenta un 7,6% por hora de retraso en la administración de los antibióticos correctos.

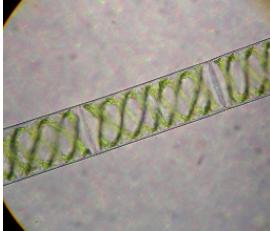
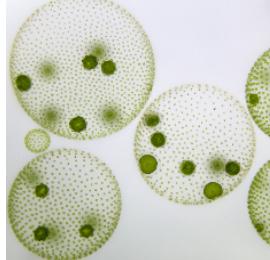
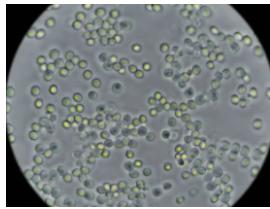
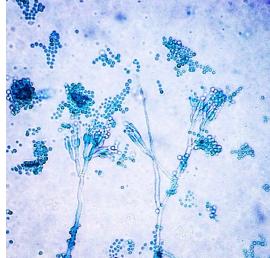
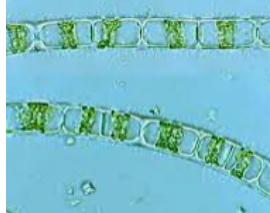
Desgraciadamente, se tarda más de 24 horas en cultivar las bacterias obtenidas de la sangre de un paciente infectado, identificar la especie y luego determinar a qué antibióticos es resistente el organismo, lo que conlleva una tasa de mortalidad muy elevada en este tipo de infecciones.

La resistencia bacteriana a los antibióticos se está convirtiendo en una importante amenaza para la salud, y la rápida identificación de las bacterias resistentes a los antibióticos es esencial para salvar vidas y reducir la propagación de la resistencia a los antibióticos.

Queremos diferenciar entre distintos tipos de micro formas de vida: Spirogyra, Volvox, Pithophora, Yeast, Raizopus, Penicillium, Aspergillus sp, Protozoa, Diatom y Ulothrix.

Contenido:

El dataset contiene 30.527 casos con 24 parámetros(columnas) que se corresponden con uno de los tipos de microbios a reconocer.

Spirogyra		Volvox	
Pithophora		Yeast	
Rhizopus		Penicillium	
Aspergillus sp		Protozoa	
Diatom		Ulothrix	

Datos de cada forma de vida(columnas):

Parámetro	Descripción
Solidity	Es la relación entre el área de un objeto y el área de un casco convexo del objeto. Se calcula como Área/Área convexa.
Eccentricity	La excentricidad es la relación entre la longitud del eje mayor y el eje menor de un objeto.
EquivDiameter	Diámetro de un círculo con la misma área que la región.
Extrema	Puntos extremos de la región. El formato del vector es [arriba-izquierda arriba-derecha arriba-derecha abajo-derecha abajo-izquierda abajo-izquierda arriba-izquierda].
Filled Area	Número de píxeles en Filled Image, devuelto como un escalar.
Extent	Relación del área de píxeles de una región con respecto al área del cuadro delimitador de un objeto.
Orientation	La dirección general de la forma. El valor oscila entre -90 grados y 90 grados.
Euler Number	Número de objetos en la región menos el número de agujeros en esos objetos.
Bounding Box 1, 2, 3 y 4	Posición y tamaño de la caja más pequeña (rectángulo) que limita el objeto.
Convex Hull 1, 2, 3 y 4	Área que envuelve a todo el objeto.
Major Axis Length	Longitud del eje del objeto de mayor tamaño.
Minor Axis Length	Longitud del eje del objeto de menor tamaño.
Perimeter	Perímetro del objeto.
Convex Área	Área convexa del objeto.
Centroid 1 y 2	Centroide del objeto.
Área	Área del objeto.
Raddi	Radio del objeto.
Microorganisms	Nomenclatura del microorganismo.

Preprocesamiento

En cuanto al preprocesamiento del dataset se han realizado las siguientes tareas:

- **Normalización:**

Se ha realizado la normalización de los atributos con los valores de media y desviación típica cero y uno respectivamente.

- **Categorización:**

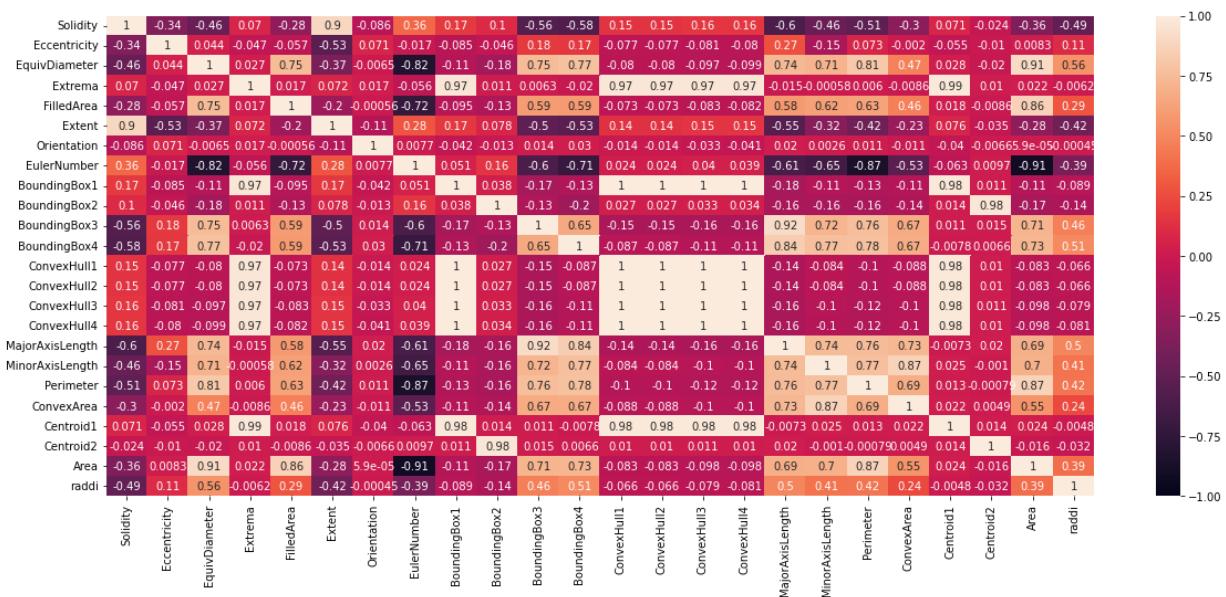
Se ha realizado una categorización del conjunto de etiquetas("labels") en modo *one hot*. El conjunto pasará de ser el índice de cada forma de vida, por ejemplo, cero la etiqueta correspondiente a la *Spirogyra*, a un vector de diez posiciones a valor cero exceptuando el índice correspondiente al valor de la etiqueta de la forma de vida, el cual se encontrará a uno.

- **Eliminación de outliers:**

Se han eliminado los casos atípicos, es decir, los que se desvían lo suficiente del conjunto de la media. Esto se ha hecho calculando el zscore a través de `scipy.stats.zscore` y eliminando los que sobrepasen 3, resultando en la eliminación de 1755 casos (aproximadamente un 5% del total original).

- **Correlaciones:**

Se ha obtenido la matriz de correlaciones para observar las relaciones entre las variables de cada fila del dataset.



Se puede observar como hay una correlación, por ejemplo, entre las variables *Centroid1* y *Extrema* con valor de 0.99. Cuando se encuentre esta correlación entre dos variables se eliminará uno de los atributos para minimizar la complejidad de los cálculos.

Cabe destacar que en este dataset no encontramos valores nulos o no válidos(**"missing values"**). Por otro lado, no se ha realizado la **bucketización**, ya que, debido a la naturaleza del problema en cuestión, no es de utilidad agrupar categorías.

Dataset final

Una vez realizado todo el preprocesamiento el dataset contiene 28.772 filas(casos) y 18 columnas(*atributos*). En el entorno de programación existirán las variables **X** e **y**, donde **X** tendrá forma de matriz con dimensiones (28.772, 17) e **y** será otra matriz con dimensiones (28.772, 1) donde se almacenarán las etiquetas correspondientes a cada caso en **X**. También existirá una variable denominada **y_onehot** donde se encontrará la etiqueta de cada caso en codificación one hot.

Subconjuntos de entrenamiento, validación y test

El dataset se ha separado en los siguientes subconjuntos:

- Entrenamiento(65%): este subconjunto se utilizará para entrenar los diferentes métodos a utilizar aprendidos durante la asignatura.
- Validación(22.5%): este subconjunto se utilizará como método de comprobación para observar si la fase de entrenamiento se ha realizado correctamente y se obtienen unos resultados óptimos.
- Test(12.5%): este último subconjunto se utilizará para obtener las métricas finales y observar el funcionamiento de los diferentes métodos utilizados en este proyecto. A partir de estos datos se realizarán las gráficas solicitadas expuestas en este documento.

```
print("X_train:", X_train.shape)
print("y_train:", y_train.shape)

print("X_val:", X_val.shape)
print("y_val:", y_val.shape)

print("X_test:", X_test.shape)
print("y_test:", y_test.shape)
```

```
X_train: (18701, 18)
y_train: (18701,)
X_val: (6546, 18)
y_val: (6546,)
X_test: (3525, 18)
y_test: (3525,)
```

Regresión logística

Adaptación

Se ha reutilizado el código de prácticas anteriores encapsulando las funciones de este en una clase denominada **My_reg_log** con pequeñas adaptaciones para permitir la nueva estructura.

Debido a que se va a realizar una regresión logística multi-clase se ha creado una nueva clase denominada **My_classifier_log**, la cual será la encargada de crear y entrenar 10 modelos especializados en reconocer cada una de las etiquetas que representan las formas de vida del dataset.

Dentro de la clase **My_reg_log** se encuentran las siguientes métodos:

- sigmoid: encargada de calcular la función sigmoide de un número, vector o matriz.
- func_coste_lamb: encargada de calcular la función de coste o error del modelo.
- func_gradiente_lamb: encargada de calcular el gradiente para obtener los valores de theta óptimos.
- train: encargada de utilizar la función *fmin_tnc* proporcionada por el módulo *scipy.optimize* para minimizar el coste.
- predict: encargada de realizar las predicciones dado una serie de parámetros.
- get_theta: devuelve el conjunto de *thetas* del modelo. Utilizada para debuggear.

Dentro de la clase **My_classifier_log**:

- treat_labels: convierte el conjunto de etiquetas en formato one hot a 10 vectores diferentes cada uno únicamente con la etiqueta correspondiente con valor 1.
- train: encargada de entrenar cada uno de los modelos de regresión lineal para cada etiqueta del dataset.
- accuracy: encargada de calcular el *accuracy* resultado el entrenamiento del modelo en su conjunto.
- theta: devuelve el conjunto de *thetas* para cada modelo de regresión lineal.

Ajuste

Se ha realizado una búsqueda del parámetro óptimo para lambda entre el conjunto de valores 0.01, 0.1, 1, 10 y 100. Podemos observar como el mejor valor para lambda de este conjunto se trata de 0.1 con un 0.64(64%) de *accuracy* en el subconjunto de validación. Realizando el mapeo de atributos, se ha observado que extendiendo los atributos del dataset hasta la segunda potencia se obtiene el mejor resultado.

```
lambda: 0.01 | val_accuracy: 0.6595
lambda: 0.1  | val_accuracy: 0.6392
lambda: 1.0  | val_accuracy: 0.6372
lambda: 10   | val_accuracy: 0.6332
lambda: 100  | val_accuracy: 0.5802
```

Resultados

Una vez entrenado el modelo con lambda 0.1 obtenemos un accuracy con el subconjunto de validación de 64%.

Por otro lado, los valores de *precision*, *recall*, *f1-score* y *support* son los siguientes:

	precision	recall	f1-score	support
<i>Spirogyra</i>	0.64	0.61	0.63	70
<i>Volvox</i>	0.56	0.61	0.58	490
<i>Pithophora</i>	0.87	0.14	0.24	144
<i>Yeast</i>	0.64	0.70	0.67	424
<i>Rhizopus</i>	0.84	0.92	0.88	297
<i>Penicillium</i>	0.54	0.48	0.51	128
<i>Aspergillus sp</i>	0.57	0.62	0.59	458
<i>Protozoa</i>	0.63	0.68	0.65	459
<i>Diatom</i>	0.33	0.06	0.10	211
<i>Ulothrix</i>	0.60	0.68	0.64	844
accuracy			0.62	3525
macro avg	0.62	0.55	0.55	3525
weighted avg	0.61	0.62	0.60	3525

Redes neuronales

Adaptación

Se ha reutilizado el código de prácticas anteriores encapsulando las funciones de este en una clase denominada **My_NN** con pequeñas adaptaciones para permitir la nueva estructura.

La nueva clase consta de las siguientes funciones:

- sigmoid: encargada de calcular la función sigmoide de un número, vector o matriz.
- rebuild: reconstruye las thetas a 2 matrices separadas, una para cada capa, a partir de una lista contigua de pesos usada por el minimizador.
- forward_prop_one: realiza la operación homónima para un único caso de prueba.
- forward_prop: realiza la operación homónima para un conjunto de datos
- func_coste_k: calcula el coste para un único caso.
- func_coste_m: calcula el coste para un conjunto de casos.
- train: entrena las thetas separando los casos de entrenamiento en tandas de tamaño N un número M de veces, ambas variables introducidas por parámetro.
- predict

- accuracy: calcula dicha métrica.
- backprop: realiza el back propagation, usada por el minimizador.

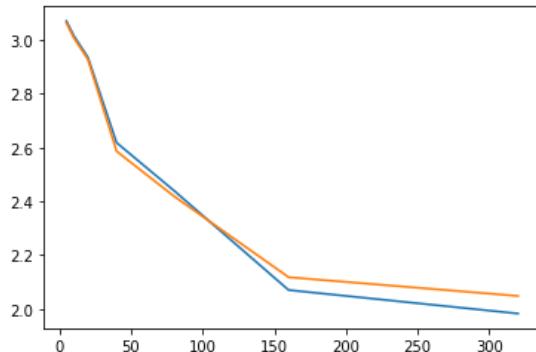
Esta clase también dispone de una función *train_old* que se corresponde a un entrenamiento siguiendo el sistema de *mini batch gradient descent*. Aunque esta parece funcionar bien en un primer momento, da un error en algún punto de la ejecución que no hemos sido capaces de identificar.

Ajuste

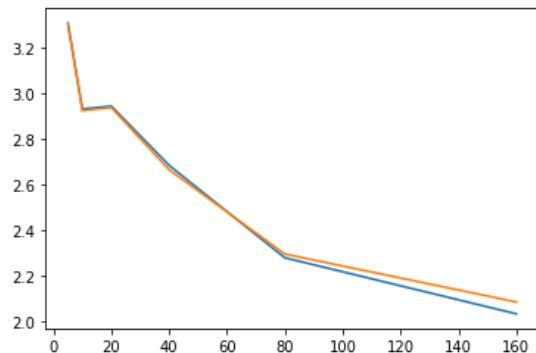
Se ha hecho una exploración de diversos hiperparametros: lambda, número de iteraciones y número de neuronas de la capa intermedia. Para hacer pruebas, se ha utilizado primero una pequeña porción de los datos de entrenamiento para agilizar el proceso:

Se han probado varios valores de lambda en los siguientes experimentos y en todos los casos lambda=1 resultaba en el mejor caso.

Coste frente a número de iteraciones para el 20% del dataset con 20 neuronas en la capa oculta:

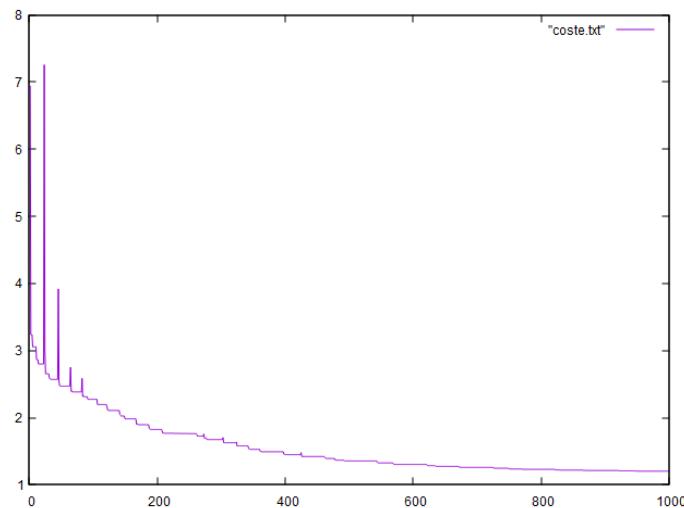


Coste frente a número de iteraciones para el 100% del dataset con 20 neuronas en la capa oculta:



El mejor resultado obtenido para 20 neuronas para la capa intermedia ha sido de 61% de accuracy. Este resultado aún es mejorable.

Se han hecho pruebas con más números de neuronas en la capa oculta y el mejor de los experimentados ha sido n=100. con un accuracy del 89%.



Resultados

La red que mejor desempeño ha mostrado ha sido la siguiente:

- Lambda = 1.
- Neuronas capa oculta = 100.
- Iteraciones = 1000.
- Tiempo de entrenamiento = 22 minutos.
- Accuracy = 82%

	precision	recall	f1-score	support											
Spirogyra	0.88	0.69	0.77	187	0	8e+02	0	1	3	0	0	1	1	0	4
Volvox	0.85	0.73	0.79	1323	1	3e+02	7e+02	0	2	0	2	13	4	15	18
Pithophora	0.93	0.60	0.73	370	2	9e-03	32.2e+02	1	0	0	11	4	12	26	
Yeast	0.91	0.78	0.84	1152	3	9e+02	14	0	9e+02	0	3	8	6	7	29
Rhizopus	0.93	0.96	0.94	803	4	33	0	0	0	7.7e+02	0	0	0	0	0
Penicillium	0.91	0.43	0.58	376	5	5e+02	0	0	0	0	1.6e+02	51	5	0	10
Aspergillus sp	0.81	0.56	0.67	1252	6	4e+02	43	0	15	45	8.68e+02	30	4	28	
Protozoa	0.91	0.91	0.91	1220	7	1e+02	0	0	7	0	8.11e+03	0	0	0	
Diatom	0.78	0.49	0.60	570	8	2e+02	18	10	14	0	0	0	3.27e+02	30	
Ulothrix	0.86	0.73	0.79	2242	9	3e+02	89	5	42	7	1	36	29	21	6e+03
micro avg	0.87	0.72	0.79	9495	0	1	2	3	4	5	6	7	8	9	
macro avg	0.88	0.69	0.76	9495											
weighted avg	0.87	0.72	0.79	9495											
samples avg	0.71	0.72	0.71	9495											

Supported vector machine(SVM)

Adaptación

Se ha reutilizado el código de prácticas anteriores encapsulando las funciones de este en una clase denominada **My_SVM** con pequeñas adaptaciones para permitir la nueva estructura.

Las variables *C* y *sigma* han pasado a formar parte de los atributos de la clase que posteriormente se utilizarán en los métodos de esta. Se ha utilizado un SVM

La clase consta de las siguientes funciones:

- *fit*: Se entrena el modelo respecto a un *X* y un *y*.
- *predict*: Dado un *X*, devuelve las etiquetas predichas para cada uno de los casos.
- *accuracy*: calcula la métrica *accuracy*.
- *find_params*: Dados los sets de entrenamiento y validación, encuentra el sigma y el *C* mejores, entrenando para cada combinación de una lista de posibles un nuevo modelo y evaluándolo usando la función *accuracy*.

Ajuste

Se ha realizado una búsqueda de los valores óptimos para *C* y *sigma* utilizando una lista con los valores 0.1, 1, 5 y 10. Probando con todas las combinaciones posibles se ha encontrado que los valores 1 y 0.1 para *C* y *sigma* respectivamente son los que mejor rendimiento presentan.

```
C: 0.1 | sigma: 0.1 | accuracy: 0.4674610449129239
C: 0.1 | sigma: 1 | accuracy: 0.7137183012526734
C: 0.1 | sigma: 5 | accuracy: 0.49495875343721357
C: 0.1 | sigma: 10 | accuracy: 0.425756186984418
C: 1 | sigma: 0.1 | accuracy: 0.9790711885120684
C: 1 | sigma: 1 | accuracy: 0.9433241674304919
C: 1 | sigma: 5 | accuracy: 0.5892147876565842
C: 1 | sigma: 10 | accuracy: 0.5044301863733578
C: 5 | sigma: 0.1 | accuracy: 0.9790711885120684
C: 5 | sigma: 1 | accuracy: 0.9732661167124962
C: 5 | sigma: 5 | accuracy: 0.6571952337305225
C: 5 | sigma: 10 | accuracy: 0.5520928811487932
C: 10 | sigma: 0.1 | accuracy: 0.9790711885120684
C: 10 | sigma: 1 | accuracy: 0.9746410021387106
C: 10 | sigma: 5 | accuracy: 0.6914146043385273
C: 10 | sigma: 10 | accuracy: 0.5733272227314391
Best C: 1
Best sigma: 0.1
```

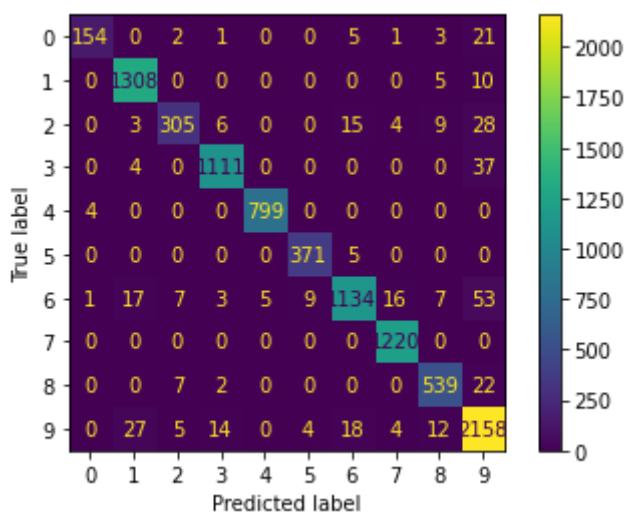
Resultados

Entrenando el modelo utilizando los valores $C = 1$ y $\sigma = 0.1$ comentados anteriormente el proceso de entrenamiento presenta un *accuracy* de 1.0(100%). Obteniendo este mismo valor con el subconjunto de validación se obtiene un *accuracy* de 0.98(98%).

Por otro lado, podemos observar los valores para *precision*, *recall*, *f1-score* y *support* utilizando la función `classification_report` gracias al módulo `sklearn.metrics`:

	precision	recall	f1-score	support
<i>Spirogyra</i>	1.00	0.80	0.89	70
<i>Volvox</i>	1.00	1.00	1.00	490
<i>Pithophora</i>	1.00	0.88	0.93	144
<i>Yeast</i>	1.00	0.98	0.99	424
<i>Rhizopus</i>	1.00	1.00	1.00	297
<i>Penicillium</i>	1.00	1.00	1.00	128
<i>Aspergillus sp</i>	1.00	0.93	0.97	458
<i>Protozoa</i>	1.00	1.00	1.00	459
<i>Diatom</i>	1.00	1.00	1.00	211
<i>Ulothrix</i>	0.92	1.00	0.96	844
<i>accuracy</i>			0.98	3525
<i>macro avg</i>	0.99	0.96	0.97	3525
<i>weighted avg</i>	0.98	0.98	0.98	3525

Si obtenemos la matriz de confusión podemos observar los siguientes resultados:



Se puede observar como la etiqueta 9 correspondiente a la forma de vida *Ulothrix* es la que más problemas presenta a la hora de identificar esta a partir de los atributos del dataset.

Comparación y Conclusiones

El método que peor rendimiento presenta es la regresión logística, con un accuracy del 64% en un tiempo de 3 minutos. Pese a haber probado múltiples valores para el parámetro *lambda*, únicamente se ha podido conseguir un porcentaje alrededor de **62%**. Pensamos que esto es debido a la complejidad del problema y las limitaciones que presenta este método a la hora de afrontar este.

Por otro lado, la red neuronal, utilizando 1000 iteraciones durante un total de 22 minutos, ha conseguido alcanzar un valor para el accuracy cerca del **82%**. Pensamos que este resultado se puede mejorar empleando más tiempo en la fase de aprendizaje del modelo. Este método es muy flexible a la hora de afrontar problemas de generalización de esta magnitud, cualidad destaca comparándolo con el método de regresión logística.

Por último, el mecanismo de Supported vector machine ha proporcionado los mejores resultados en un tiempo considerablemente menor a la red neuronal, siendo estos de un accuracy del **98%** en un intervalo de tiempo de 30 segundos. Este gran rendimiento por parte del SVM se debe a la gran capacidad de tratar con datasets de gran dimensionalidad y flexibilidad a la hora de modelar diversas fuentes de información.

En conclusión, la SVM es la que, con diferencia, mejor y más rápido funciona para la tarea de clasificar entre los microorganismos del dataset.

Bibliografía

<https://www.kaggle.com/datasets/sayansh001/microbes-dataset>

<https://stackoverflow.com/questions/23199796/detect-and-exclude-outliers-in-a-pandas-database>

<https://scikit-learn.org/stable/modules/svm.html>

<https://medium.com/@subashkharel/perceptron-vs-svm-a-quick-comparison-6b5d6b5d64f>