

Correcciones Implementación sin framework

Elisa María Bonilla Martín
A01028576

Instituto Tecnológico de Estudios Superiores de Monterrey
Campus Estado de México

1 Correcciones

Algoritmo implementado manualmente y corriendo de manera directa desde el compilador.	60 ptos. Es claro que el algoritmo es implementado al menos el 95% manualmente	30 ptos. Utiliza más del 5% del código con librerías ya implementadas	0 ptos. No cumple o no entregado
---	---	--	-------------------------------------

En mi primera entrega utilizaba pandas, numpy y sklearn. Específicamente utilizaba las siguientes funciones

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```


Para reducir mi uso de las librerías utilice las siguientes líneas de código para hacer el train test split:

```
n_train = math.floor(s_f * feat.shape[0])
n_test = math.ceil((1-s_f) * feat.shape[0])
X_train = feat[:n_train]
y_train = target[:n_train]
X_test = feat[n_train:]
y_test = target[n_train:]
```

Y para el accuracy score hice lo siguiente:

```
acc = np.sum(np.equal(X_train_pred, y_train)) / len(y_train)

print(acc)
```


Predicciones	10 ptos. Incluye datos para hacer predicciones y hace varias predicciones demostrando la efectividad de su implementación.	5 ptos. Incluye datos para hacer predicciones, pero no hace ninguna predicción que demuestre la efectividad de su implementación	0 ptos. No cumple o no entregado
			

En la primera entrega no había realizado predicciones por lo que añadí la siguiente parte:




```
X_test_pred = modelo1.pred(X_test)
print(X_test_pred.shape)

acc = np.sum(np.equal(X_test_pred, y_test)) / len(y_test)
print(acc)

#Y con el mismo modelo se hizo una prueba y el accuracy fue de 0.658008658008658
```

Pruebas con un dataset	20 ptos. Incluye dataset para entrenamiento y pruebas. Además hace varias pruebas variando los datasets de training y test.	10 ptos. Incluye dataset para entrenamiento y pruebas. Pero solamente hace una prueba.	0 ptos. No cumple o no entregado
			

En la primera entrega no había proporcionado el dataset que estaba usando pero ya lo añadí al repositorio de github y aquí está la prueba.

elisabm Add files via upload		53e9a9f 12 minutes ago 5 commits
 RegresionLogistica_0.py	Add files via upload	11 days ago
 RegresionLogistica_mejorada.py	Add files via upload	12 minutes ago
 diabetes2.csv	Add files via upload	2 hours ago

Análisis del nivel de accuracy y/o error (loss)	10 a >0 pts Realiza un breve documento en el que analiza el accuracy y/o error de su implementación variando algunos hiperparámetros	0 ptos. No cumple o no entregado
---	---	-------------------------------------

Y por último realice lo siguiente para encontrar los mejores parámetros

```
#Se crea una función que prueba los diferentes learning rates

learning_rates = [0.1, 0.15, 0.5, 0.04]
models = {}

for i in learning_rates:
    print ("learning rate es: ",i)
    models[i+1] = RegLogistica(learning_rate = i, num_i=1000)
    models[i+1].fit(X_train, y_train)
    X_train_pred = models[i+1].pred(X_train)
    acc = np.sum(np.equal(X_train_pred, y_train)) / len(y_train)
    print("El accuracy es: ",acc)
    print ("-----")

'''Se obtuvo que para
learning rate es:  0.1
El accuracy es:  0.6685288640595903
-----

learning rate es:  0.15
El accuracy es:  0.6741154562383612
-----

learning rate es:  0.5
El accuracy es:  0.6536312849162011
-----

learning rate es:  0.04
El accuracy es:  0.6480446927374302
-----

'''

#Podemos ver que para este problema no se debe de usar un learning rate tan bajo
```

