Este archivo es el código base para la parte práctica del examen del tercer parcial de la clase de Inteligencia Artificial

Instrucciones

Por favor lea cuidadosamente y codifique lo que se le pida. Las instrucciones se verán de la siguiente manera:

```
***# Este ese el formato para las instrucciones***
```

Entregables

Se entrega individualmente un archivo comprimido en Zip que contenga:

- La libreta de Colab (extensión ipynb)
- LA libreta de Colab en formato PDF; use la opción de imprimir para generarlo

Evaluación

- La celda requerida debe ejecutar sin errores: 1 pts
- La celda requerida implementa el código solicitado: 1pts

Código de ética profesional

Al entregar este archivo con sus implementaciones, acepta que el trabajo realizado es de su autoría y que de confirmarse lo contrario se anulará su examen.

```
Haga doble clic para editar la celda y llenar los datos correspondientes:
```

Nombre del estudiante: Enrique Ulises Báez Gómez Tagle

Fecha de entrega: 1/06/2023

```
### import the libraries and modules required

# libraries to manipulate the data and to visualise it
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
# this is the library that contains the NN capabilities
import sklearn
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
# the evaluation metrics for regression
from sklearn.metrics import mean_squared_error
# for hyper parameter tuning
from sklearn.model_selection import GridSearchCV
# to meassure the execution time of the neural networks
import time
```

Carga del conjunto de datos

```
Cargue la base de datos gym-1sec.csv y muestre los últimos 5 registos:
```

```
# pd.read_csv()
df = pd.read_csv(filepath_or_buffer='gym-1sec.csv', sep=',')
df.index = np.arange(1, len(df) + 1)
```

```
print('Dataset size {} columns and {} rows'.format(df.shape[1], df.shape[0]))
df.tail()
```

```
Dataset size 6 columns and 10125 rows
```

	date	pre	alt	hum	tem	occ
10121	2019-09-24 16:17:05	94960.91	543.86	56.44	28.61	M
10122	2019-09-24 16:17:06	94958.60	544.06	56.48	28.61	M
10123	2019-09-24 16:17:07	94958.60	544.06	56.72	28.63	M
10124	2019-09-24 16:17:08	94961.06	543.85	56.99	28.64	M
10125	2019-09-24 16:17:09	94958.84	544.04	56.96	28.65	M

Visualice la cantidad de datos que hay en cada una de los differentes valores de la característica (feature) nivel de occupación.

La descripción de las variables es como sigue de izquierda a derecha:

* Fecha de registro
* Presión barométrica en hecto-pascal
* Altura relativa desde el nivel del mar en metros
* Humedad relativa en porcentaje
* Temperatura en grados celcius
* Nivel de ocupación en etiquetas

```
# <mi conjunto de datos>.groupby('mi variable de interés').size()
df.groupby('occ').size()
```

```
occ
H    2358
L    2442
M    5325
dtype: int64
```

Utilice el análisis de los 5 números para visualizar la descripción del cconjunto de datos

```
# .describe()
df.describe()
```

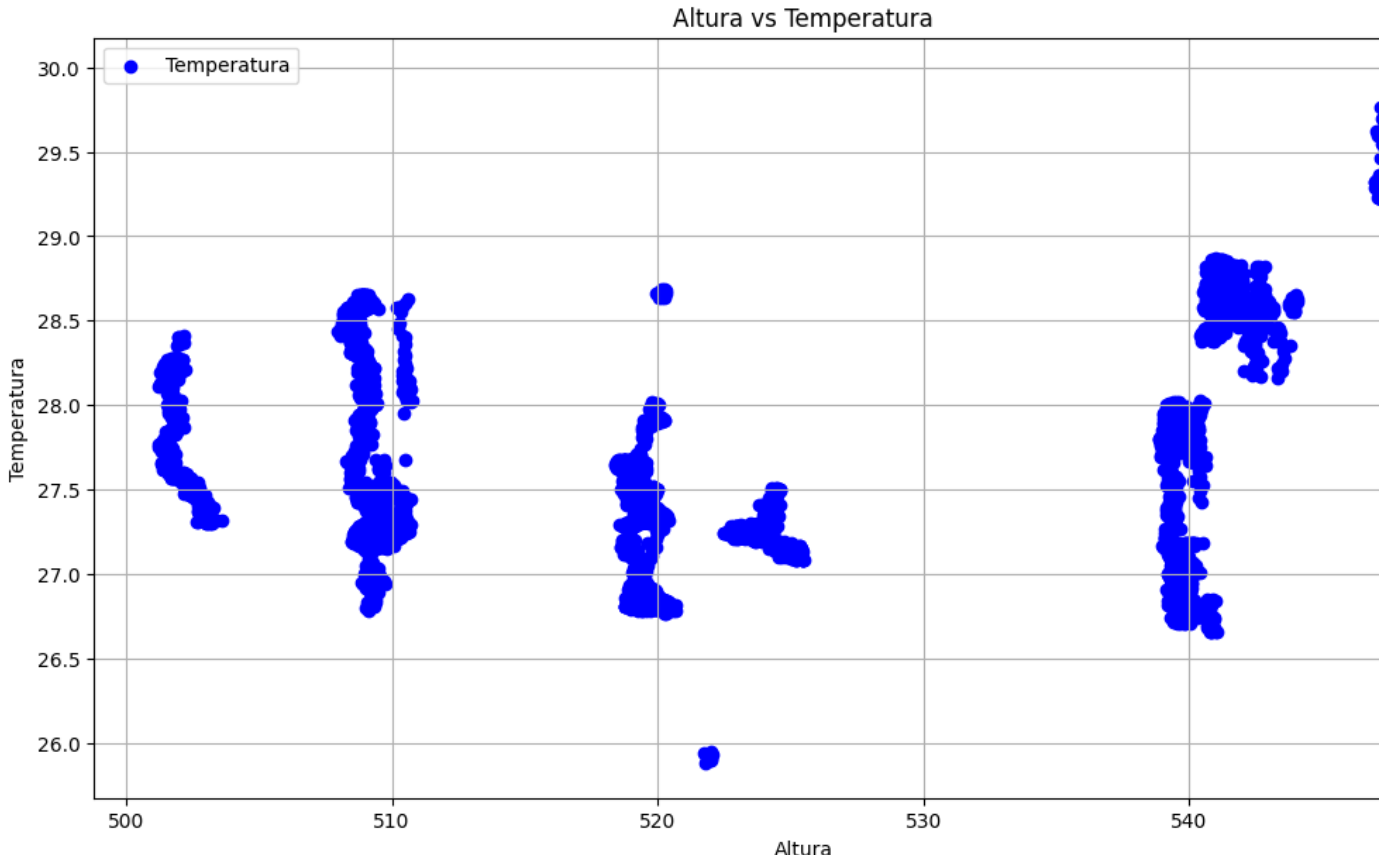
	pre	alt	hum	tem
count	10125.000000	10125.000000	10125.000000	10125.000000
mean	95171.915784	525.373692	56.746984	27.808434
std	169.404796	14.839785	2.293076	0.820957
min	94893.870000	501.210000	50.660000	25.880000
25%	95000.700000	509.410000	55.390000	27.210000
50%	95230.360000	520.240000	56.400000	27.490000
75%	95354.250000	540.370000	58.620000	28.510000
max	95447.960000	549.790000	65.390000	29.970000

Visualización de los datos

Modifique el siguiente código para visualizar la temperatura de la sala de todos los niveles de ocupación (en el eje y) respecto a la

Nombre la gráfiica adecuadamente a los datos que se muestran

```
# l = <mi conjunto de datos>[<mi conjunto de datos>.<variable de nivel de ocupación> == "valor de la variable de ocupación
# EN LUGAR DE NIVEL DE OCUPACION SI ES TEMPERATURA EN Y
y = df['tem']
x = df['alt']

fig, ax = plt.subplots()
fig.set_size_inches(13, 7)

# ax.scatter(l['variable en el eje x'], l['variable en el eje y'], label="etiqueta de los valores", facecolor="blue")
ax.scatter(x, y, label="Temperatura", facecolor="blue")

ax.set_xlabel("Altura")
ax.set_ylabel("Temperatura")
ax.grid()
# ax.set_title("Un título que corresponda con mi gráfica")
ax.set_title("Altura vs Temperatura")
ax.legend()

plt.show()
```



Generación del conjunto de datos de entrenamiento y testeo

Utilice la siguiente función para hacer la codificación a formato numérico de los diferentes valores de la variable de interés

```
def coder(feature):
    if feature == 'L':
        return 0
    elif feature == 'M':
        return 1
    else:
        return 2

# <mi conjunto de datos>['variable de interés'] = <mi conjunto de datos>['variable de interés'].apply(coder)
df['occ'] = df['occ'].apply(coder)
```

Genere los conjuntos de datos de entrenamiento y testeo de la siguiente manera:

* El conjunto de testeo debe ser el 5% aleatorio del conjunto de datos total
* Para el conjunto de entrenamiento, utilice todas las variables/características del conjunto de datos con excepción de: 'date' y 'occ'
* Seleccione la varible de nivel de ocupación como la variable/característica de interés para hacer la regresion

```
# X will be our dataset without the feature of interest
# X = <mi conjunto de datos>.drop(['mis variables que no son de interés'], axis=1)
X = df.drop(['date', 'occ'], axis=1)

# converting into numpy array and # FIXME: assigning petal length and petal width (not about flowers)
X = np.array(X)

# y will be our dataset with the feature of interest
# y = <mi conjunto de datos>['mi variable de interés']
Y = df['occ']

# VALIDATE SHAPES
print('X shape: {}'.format(X.shape))
print('Y shape: {}'.format(Y.shape))

# Splitting into train and test
# <mis subconjuntos de entrenamiento y testeo> = train_test_split(X,y,test_size=0.5,random_state=42)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.05, random_state=42)
```

```
X shape: (10125, 4)
Y shape: (10125,)
```

Escalamiento

Si es necesario escalar los datos hágalo; en caso contrario, escribir no es necesario escalar los datos en la celda como comentario

```
print('Min values of the dataset are: \n{}'.format(df.min()))
print('Max values of the dataset are: \n{}'.format(df.max()))

# Determine if the data needs to be scaled
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

print('Min values of the dataset are: \n{}'.format(X_train_scaled.min(axis=0)))
print('Max values of the dataset are: \n{}'.format(X_train_scaled.max(axis=0)))

print('Min values of the dataset are: \n{}'.format(X_test_scaled.min()))
print('Max values of the dataset are: \n{}'.format(X_test_scaled.max()))
```

```
Min values of the dataset are:
date    2019-09-18 19:04:00
pre                94893.87
alt                  501.21
```

```
hum                  50.66
tem                  25.88
occ                      0
dtype: object
Max values of the dataset are:
date    2019-10-02 20:48:58
pre                95447.96
alt                  549.79
hum                   65.39
tem                   29.97
occ                       2
dtype: object
Min values of the dataset are:
[-1.64809243 -1.629142   -2.64754226 -2.35274138]
Max values of the dataset are:
[1.63033299 1.65211843 3.76547249 2.63974148]
Min values of the dataset are:
-2.328328262015973
Max values of the dataset are:
3.0384029274199404
```

Regresion

Definición del modelo

Defina un modelo de red neuronal con las siguientes características:

* 4 capas en total
* La capa de salida debe tener el mimso número de neuronas que el número de resultados esperados por cada renglón/predicción
* La capa de entrada debe tener el mismo número de neuroas que el número de niveles de ocupacón
* La primer capa intermedia debe tener el número de neuronas igual al número de características/variables del conjunto de datos de e
* La segunda capa intermedia debe tener la mitad de neuronas que la capa anterior
* Un máximo número de iteraciones del número de clases/tipos de nivel de ocupación multiplicado por 100
* Usar la función de activación tangente hiperbólica
* Usar el solucionador adam

```
# MLPRegressor(
#     activation = 'adam',
#     solver = 'tanh'
# )

from sklearn.neural_network import MLPRegressor

# Number of features and expected results
feature_num = X_train.shape[1]
exp_res_num = len(Y_train.unique())

mlp_reg = MLPRegressor(hidden_layer_sizes=(feature_num, feature_num // 2),
                       activation='tanh', solver='adam',
                       max_iter=exp_res_num * 100, random_state=42)
```

Entrenamiento

Entrene el modelo definido con anterioridad y muestrer su tiempo de ejecución en segundos

Importante: esto puede tomar aproximadamente unos 5 minutos

```
# <mi modelo de regresión>.fit(<subconjunto de entrenamiento en x>, <subconjunto de entrenamiento en y>)
start_time = time.time()
mlp_reg.fit(X_train_scaled, Y_train)
end_time = time.time()
```

```
print('Time elapsed on training: {:.2f} seconds'.format(end_time - start_time))
# 5.708560466766357
```

```
Time elapsed on training: 9.37 seconds
```

Testeo y evaluación del modelo

```
Utilice el modelo entrenado para hacer la regresión de los datos de testeo
```

```
# <mi modelo de regresión>g.predict()
predictions = mlp_reg.predict(X_test_scaled)
print(predictions)
# array([1.99312555, 0.32425411, 1.03872412, ..., 1.956635  , 0.26321265,
# 1.17075748])
```

```
 9.37922342e-01  1.94396608e+00  9.80068229e-01  1.87607759e+00
-3.29507504e-02  1.92190227e+00  7.53364146e-01  2.04783728e+00
 1.25893201e-01  9.80598540e-01  9.79177857e-01  1.97834942e+00
 9.58621705e-01  1.05470807e+00  1.05104621e+00  1.04351418e+00
 9.34066486e-01  2.83145347e-02  2.05446683e+00 -2.03394630e-02
-2.72424042e-02  1.05277448e+00 -2.25664539e-02  9.82180543e-01
 1.05638854e+00  9.67462444e-01  7.00500071e-02  1.00751283e+00
 1.00190382e+00 -1.75382722e-02  9.71727852e-02  1.00200681e+00
 1.03068245e+00  9.98498090e-01  8.92668423e-01  1.05088780e+00
 1.03971214e+00  1.51562345e-01  1.05337109e+00  2.14943629e+00
 1.66572015e+00 -2.25595749e-02  8.51123469e-01  2.06226518e+00
 1.00907416e+00  2.04708242e+00  9.98260741e-01  1.43762038e+00
 6.67235930e-03  1.05545032e+00 -1.81653635e-02  9.32193435e-01
 1.05208231e+00  1.03674463e+00  2.16419960e+00 -2.23113989e-02
 1.73958647e+00  1.17504807e+00  9.79435131e-01  9.91722328e-01
 1.04283875e+00  2.07700910e-01  9.79110228e-01  1.01578782e+00
 9.81704299e-01  2.10040534e+00  2.03697720e+00 -2.29044003e-02
 3.08719146e-01  1.05340175e+00  1.05592223e+00  2.04857494e+00
 8.90856963e-01  1.92147241e+00  1.05325316e+00  1.08904242e+00
 1.05809106e+00  9.78990647e-01  1.66836782e+00  1.91187643e+00
 1.88942994e+00  9.78632152e-01  9.90480114e-01  1.89008963e+00
 1.05421841e+00 -5.64131420e-02 -2.03098167e-02  1.10906477e-01
 9.79248761e-01  9.84774125e-01  1.02795848e+00  1.05268310e+00
 9.81026432e-01  1.00243698e+00  4.44022773e-01  7.48178475e-01
 1.00585364e+00  2.05554804e+00  9.49176339e-01  1.01250067e+00
 8.20659568e-02 -1.18202471e-02  1.05087434e+00  2.04794349e+00
 2.12906455e+00  6.83677704e-02  9.21706569e-01  8.91483041e-01
 1.10069823e+00  9.55651569e-01  1.87991473e+00  2.05648079e+00
 1.93455279e+00 -8.95209206e-03  9.27214643e-01  1.92819148e+00
 9.79913131e-01  1.85664051e+00  1.91948858e+00  2.05174975e+00
 2.07455642e+00 -5.02116595e-02  9.80612486e-01  9.79240294e-01
 3.39340823e-02  9.52700903e-01  1.34968516e+00  9.33098182e-02
 2.15795424e+00  1.75071246e+00  2.61963648e-02 -4.49260962e-02
 9.77073615e-01  9.88066589e-01  9.81120923e-01  9.13482779e-01
 1.27545716e+00  1.82310855e+00  1.04157489e+00  1.43697530e-02
 1.90991043e+00  1.04169991e+00  9.83761628e-01  1.05369940e+00
 9.25307992e-01  1.13787966e-01  2.07309185e+00  1.05037592e+00
 1.05787397e+00  1.14834555e+00  9.41379351e-01  1.04444754e+00
 1.05558940e+00  9.80206759e-01  9.40121153e-01  2.17443750e+00
 1.05482513e+00  1.05352270e+00  1.03564343e+00  2.14386635e+00
 9.79077616e-01  2.05365218e+00  1.00773216e+00  2.14732890e+00
 9.79995885e-01  1.05830770e+00  9.79926988e-01  2.10248401e+00
 2.07047494e+00 -3.37149563e-02  9.64593340e-01  1.05385677e+00
 1.05324432e+00  9.79156325e-01  1.05522128e+00  1.02979041e+00
 1.92008331e+00  1.02461195e+00  1.86341108e+00  9.68178527e-01
 9.66630782e-01  1.77816932e+00  1.00079533e+00  1.41682120e+00
 2.05288393e+00  1.04499436e+00  1.71865950e+00  1.95087183e+00
 1.04301347e+00  1.05753010e+00  2.08102411e+00 -2.87429395e-02
 1.93149201e+00  9.79192795e-01  9.10686103e-01  1.93391664e+00
 2.04732517e+00  1.10971342e+00  8.49839041e-01  9.80352158e-01
 1.05812887e+00  1.05101700e+00  1.05371695e+00  1.48769012e+00
 2.05780034e+00  1.00283672e+00  2.12200725e+00  1.11895522e+00
 1.04703849e+00  1.86806368e+00  1.43160622e+00  9.59270241e-01
 8.98141236e-01  9.15717966e-01  1.06193190e+00  2.04287983e+00
 1.01718726e+00  2.10847286e+00  9.79747354e-01 -3.96338389e-03
 2.57242534e-01  1.05496611e+00  5.21167381e-02  9.62098206e-01
 9.57456020e-01  4.45990495e-01  8.94443114e-01  9.85540166e-01
 1.43974160e+00 -3.65211765e-02  1.83702445e+00]
```

Evalue el mejor modelo de regresión para mostrar su error cuadrático medio

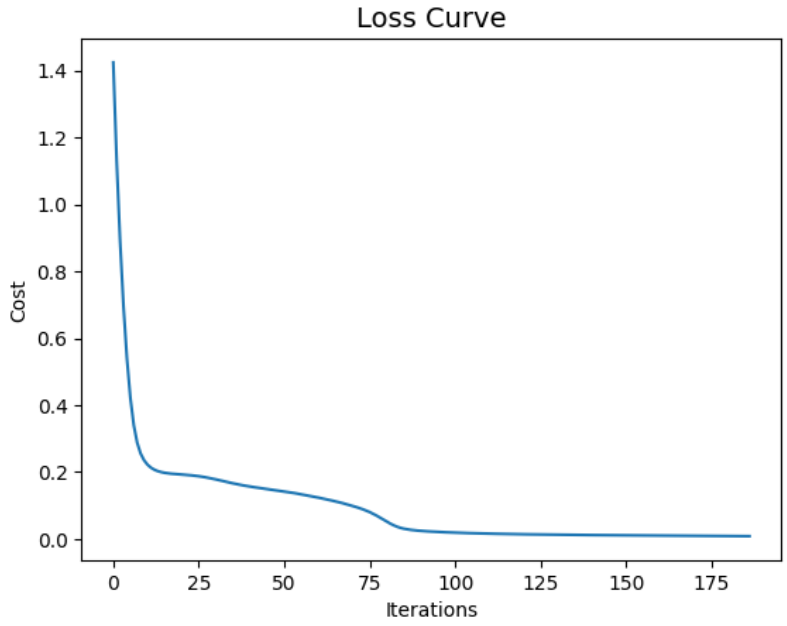
```
# print('mse: {:.2f}'.format(mean_squared_error(<subconjunto de testeo en y>, <predicciones realizadas por el modelo>)))
print('mse: {:.2f}'.format(mean_squared_error(Y_test, predictions)))
# mse: 0.03
```

```
mse: 0.02
```

Visualice la curva de entrenamiento

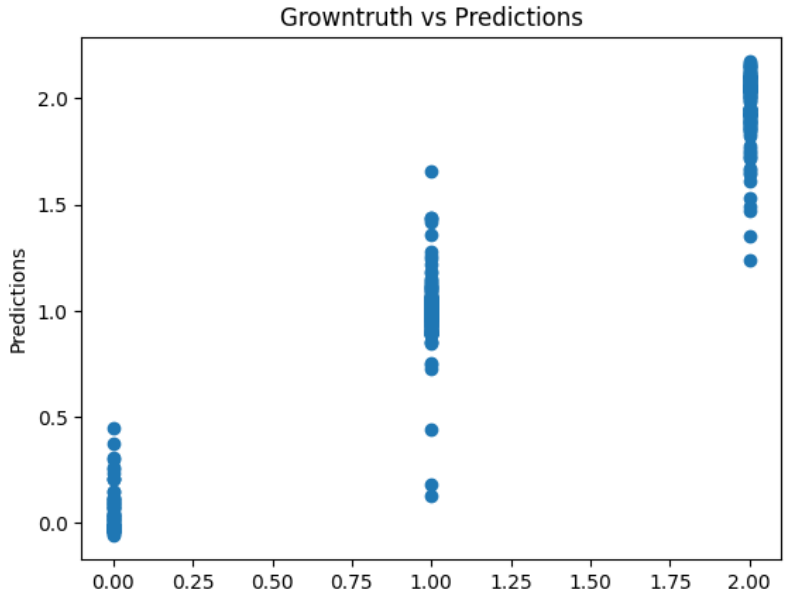
```
# see how well was the training of the model
# plt.plot(<mi modelo de regresión>.loss_curve_)

# Show accuracy rate
print('Accuracy rate: {:.2f}'.format(mlp_reg.score(X_test_scaled, Y_test)))

plt.plot(mlp_reg.loss_curve_)
plt.title("Loss Curve", fontsize=14)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.show()
```

```
Accuracy rate: 0.96
```



Visualice la diferencia entre los valores reales y los predichos por la red neuronal

```
# pd.DataFrame({'growntruth': <subconjunto de testeo en y>,'predictions': <predicciones>})
pd.DataFrame({'growntruth': Y_test, 'predictions': predictions})
# Plot this comparison as scatter plot
plt.scatter(Y_test, predictions)
plt.xlabel('Growntruth')
plt.ylabel('Predictions')
plt.title('Growntruth vs Predictions')
plt.show()
```



Con el valor del accuracy y las gráficas de entrenamiento y matriz de confusión explique brevemente el rendimiento del modelo de red

We got an accuracy of 0.96 which is really good, and the loss curve shows that the model is learning. From the plot we can see that there is a correct estimation of data

Búsqueda de hiperparámetros a través de una matriz

Defina una matriz de búsqueda de hiperparámetros que contenga lo siguiente:

* Un modelo de red neuronal con el número de capaz y neuronas que considere lleve a mejorar el aprendizaje en la tarea de regresión
* El número de iteraciones que consideren sean pertinentes
* El optimizador de la familia de los métodos quasi-Newton
* La función de activación relu

```
param_grid = {
    'activation': ['relu'],  # Function of activation
    'hidden_layer_sizes': [(64, 128, 64)],  # Number of neurons in the hidden layers
    'max_iter': [100, 200, 300],  # Maximum number of iterations
    'solver': ['lbfgs', 'bfgs', 'newton-cg']  # Quasi-Newton optimizer
}
```

Defina la búsqueda de la matriz sin validación cruzada

```
grid_search = GridSearchCV(mlp_reg, param_grid, n_jobs=-1)
```

Execute la búsqueda de los hiperparámetros y muestre el tiempo total de execución

Importante: tenga en mente que esta sección puede tardar varios minutos, de acuerdo a los modelos incluídos en la búsqueda

```
tc_start = time.time()
grid_search.fit(X_train_scaled, Y_train)
tc_end = time.time()
# print(tc_end-tc_start)
print('Time elapsed on training: {:.2f} seconds'.format(tc_end - tc_start))
# 13.170661687850952
```

Muestre el 'mejor modelo' de la búsqueda de los hiperparámetros

```
# print('The best hyper parameter values are:\n{}'.format(<mi búsqueda de cuadrícula>.best_params_))
print('The best hyper parameter values are:\n{}'.format(grid_search.best_params_))
# The best hyper parameter values are:
# {'activation': 'tanh', 'hidden_layer_sizes': (8, 16, 8), 'max_iter': 200, 'solver': 'lbfgs'}
```

Obtenga y muestre las predicciones con el mejor modelo de la búsqueda de hiperparámetros

```
# <mi búsqueda de cuadrícula>.best_estimator_.predict()
predictions = grid_search.best_estimator_.predict(X_test_scaled)
# array([2.01497909, 0.10336705, 1.00173473, ..., 1.99163308, 0.00839881,
# 0.987649  ])
predictions
```

Evalue el mejor modelo de regresión resultante de la búsqueda de hiperparámetros para mostrar su error cuadrático medio

```
# print('mse: {:.2f}'.format(mean_squared_error()))
print('mse: {:.2f}'.format(mean_squared_error(Y_test, predictions)))
# mse: 0.00

# Show new loss curve + accuracy
print('Accuracy rate: {:.2f}'.format(grid_search.best_estimator_.score(X_test_scaled, Y_test)))
plt.plot(mlp_reg.loss_curve_)
plt.title("Loss Curve", fontsize=14)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.show()
```

Visualice la diferencia entre los valores reales y los predichos por la red neuronal

```
# pd.DataFrame({'growntruth': <subconjunto de testeo en y>,'predictions': <predicciones>})
pd.DataFrame({'growntruth': Y_test, 'predictions': predictions})
# Plot this comparison as scatter plot
plt.scatter(Y_test, predictions)
plt.xlabel('Growntruth')
plt.ylabel('Predictions')
plt.title('Growntruth vs Predictions')
plt.show()
```

Conclusiones

De acuerdo a los resultados de las métricas de evaluación y a las gráficas de matriz de confusión, mencione qué modelo de red neuronal

**With these hypertuned parameters, a rate of 1 was achieved, which says that our model was overtrained with these parameters found. **

Executing (4s) <cell line: 2> › fit() › _run_search() › evaluate_candidates() › __call__() › __call__() › retrieve() › wrap_future_result() › result() › wait()