Problema #1: Regresión

Metodología técnica

Primero importamos librerías, cargamos el csv y limpiamos los datos nulos:

Importar librerias

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_percentage_error
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
```

✓ 0.0s Python

Cargar el csv en un dataframe

```
regresion_df = pd.read_csv("ProblemaRegresion.csv")
print(regresion_df)
```

✓ 0.1s Python

Verificar los datos nulos

```
regresion_df.isna().any()
```

✓ 0.0s Python

Limpiar los datos nulos

```
regresion_df.dropna(inplace=True)
```

✓ 0.1s Python

Posteriormente generamos una matriz de correlación a través de un heatmap.

Examen medio Curso Redes neuronales Abner Zair Velazquez Domínguez 1960233

Generar una matriz de correlacion para identificar cuales podemos emplear para simplificar nuestro modelo

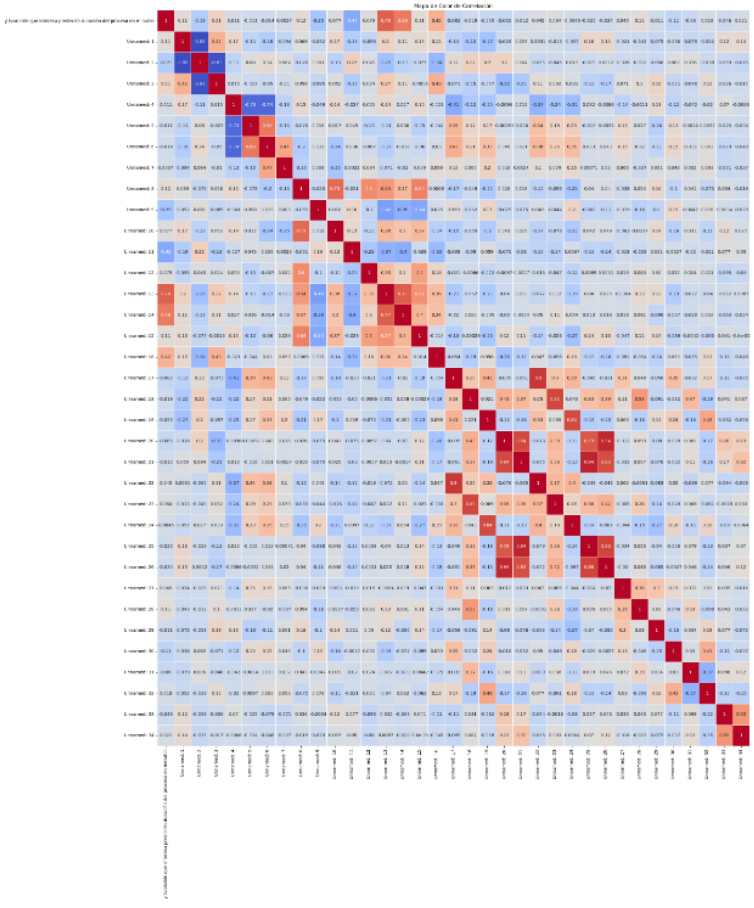
```
correlation_matrix = regresion_df.corr()
```

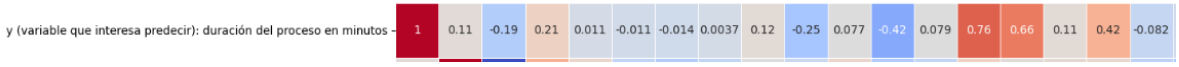
✓ 0.2s Python

```
plt.figure(figsize=(30, 30))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Mapa de Calor de Correlación')
plt.show()
```

✓ 4.1s Python

Esta matriz es importante para obtener los valores en X quien nuestro modelo de regresión usaremos.





Pude observar que los valores con puntos mas separados del 0 son los 11,13,14 y 16. Por lo tanto hice pruebas con ellos. Específicamente con los arreglos de [13,14], [11,13,14], [13,14,16] y [11,13,14,16]. Me quedé con este último.

Identificamos que el 11 y 16 tiene grado +-.42 mientras que el 13 y 14 tienen grado .76 y .66 respectivamente

```python
ncol = regresion_df.shape[1]
X = regresion_df.iloc[:, [11,13,14,16]]
Y = regresion_df.iloc[:, 0]
print(X)
```

Python

Generamos el modelo de regresion lineal para predecir valores en Y

```python
model = MLPRegressor(solver='adam', hidden_layer_sizes=(200,), verbose=True,max_iter
model.fit(X, Y)
pred = model.predict(X)
#print(y,pred)
print(X)
#print(y)
```

3.9s

Python

Aquí podemos ver los resultados, cabe mencionar que utilice la función de activación relu a 1000 iteraciones maximas:

```
Iteration 1, loss = 619.91906471
Iteration 2, loss = 475.17437662
Iteration 3, loss = 475.95455477
Iteration 4, loss = 408.89484050
Iteration 5, loss = 426.75202871
Iteration 6, loss = 408.46574473
Iteration 7, loss = 402.14006865
Iteration 8, loss = 401.58481861
Iteration 9, loss = 394.03735770
Iteration 10, loss = 394.67348804
Iteration 11, loss = 380.87948092
Iteration 12, loss = 377.11907647
Iteration 13, loss = 374.21333321
Iteration 14, loss = 363.22969729
Iteration 15, loss = 367.68435444
Iteration 16, loss = 364.19266219
Iteration 17, loss = 361.88218941
Iteration 18, loss = 354.29602591
Iteration 19, loss = 352.23640864
Iteration 20, loss = 340.83449123
Iteration 21, loss = 338.03954981
Iteration 22, loss = 337.12813073
Iteration 23, loss = 328.22372016
Iteration 24, loss = 326.62671937
Iteration 25, loss = 323.53023447
```

Presentamos el error porcentual absoluto y convertimos esto a presicion porcentual en el conjunto de entrenamiento

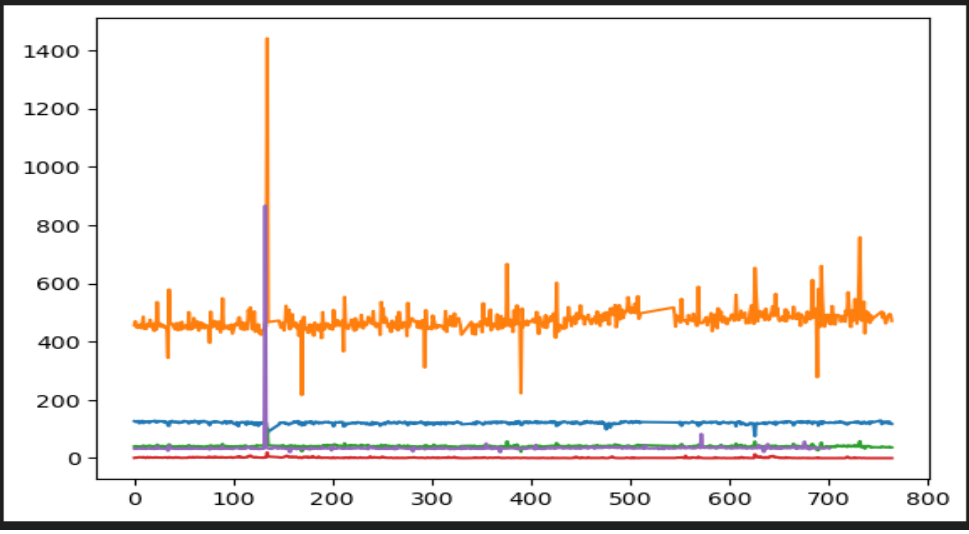
```
mape = mean_absolute_percentage_error(Y, pred)

print('MAPE:',mape)
print('Precisión porcentual en el conjunto de entrenamiento:', (1-mape)*100.0)


plt.plot(X)  # Mostrar solo los valores de 100 a 200 en el eje x
plt.plot(pred)
plt.show()
```

0.3s Python

```
MAPE: 0.03543100562985681
Precisión porcentual en el conjunto de entrenamiento: 96.45689943701431
```



Resultados

En nuestras condiciones utilizando los valores unnamed 11,13,14 y 16 se obtuvo una precisión porcentual en el conjunto de entrenamiento de 96 y un *mean_absolute_percentage_error de* .007 aproximadamente.

Con todos los datos (sin contar los eliminados por las filas con datos faltantes), se obtiene una precisión mayor a cambio de eficiencia. Queda a consideración.

Conclusiones

Se concluyó que:

- Utilizar todos los valores en X, si bien no es eficiente, genera en esta ocasión (con las filas incompletas eliminadas, a 1000 iteraciones y con la función de activación relu) una precisión muy alta.
- En el caso más ideal de nuestro procedimiento, las variables unnamed 11, 13, 14 y 16 son las que tienen mayor correlación
- Con dichos valores unnamed, se obtiene una eficiencia de 96 puntos y un *mean_absolute_percentage_error de .03 aproximadamente*

Problema 2: Mnist

Metodología técnica

El problema se divide en 2:

- Mnist4mlp.by: Este archivo es como una interfaz que emplea 2 archivos .sav, uno con un modelo entrenado de regresión lineal y otro con un modelo entrenado de regresión logística.
- Mnist2b.py: Este archivo genera los dos archivos .sav necesarios para correr el programa principal.
- 2 archivos .csv utilizadas para los procesos de entrenamiento de mnist2be imágenes ejemplo.

No veo necesidad de explicar línea a línea los archivos ya dados debido a que viene explicados por comentarios. Los añadidos son los siguientes:

Entrenas un nuevo modelo pero esta vez de regresión logística (en el archivo mnist2b.py)

```
# Entrenar un modelo de regresión logística
logistic_reg_model = LogisticRegression()
logistic_reg_model.fit(X, y)
```

Predecimos los valores de X_test anteriormente escritos

```
# predecir los valores de X_test
predicted = logistic_reg_model.predict(X_test)
```

Calculamos el error, esto lo hacemos porque es inverso a los puntos de accuracy

```
# para finalizar se calcula el error
error = 1 - accuracy_score(y_test, predicted)
print(error)
```

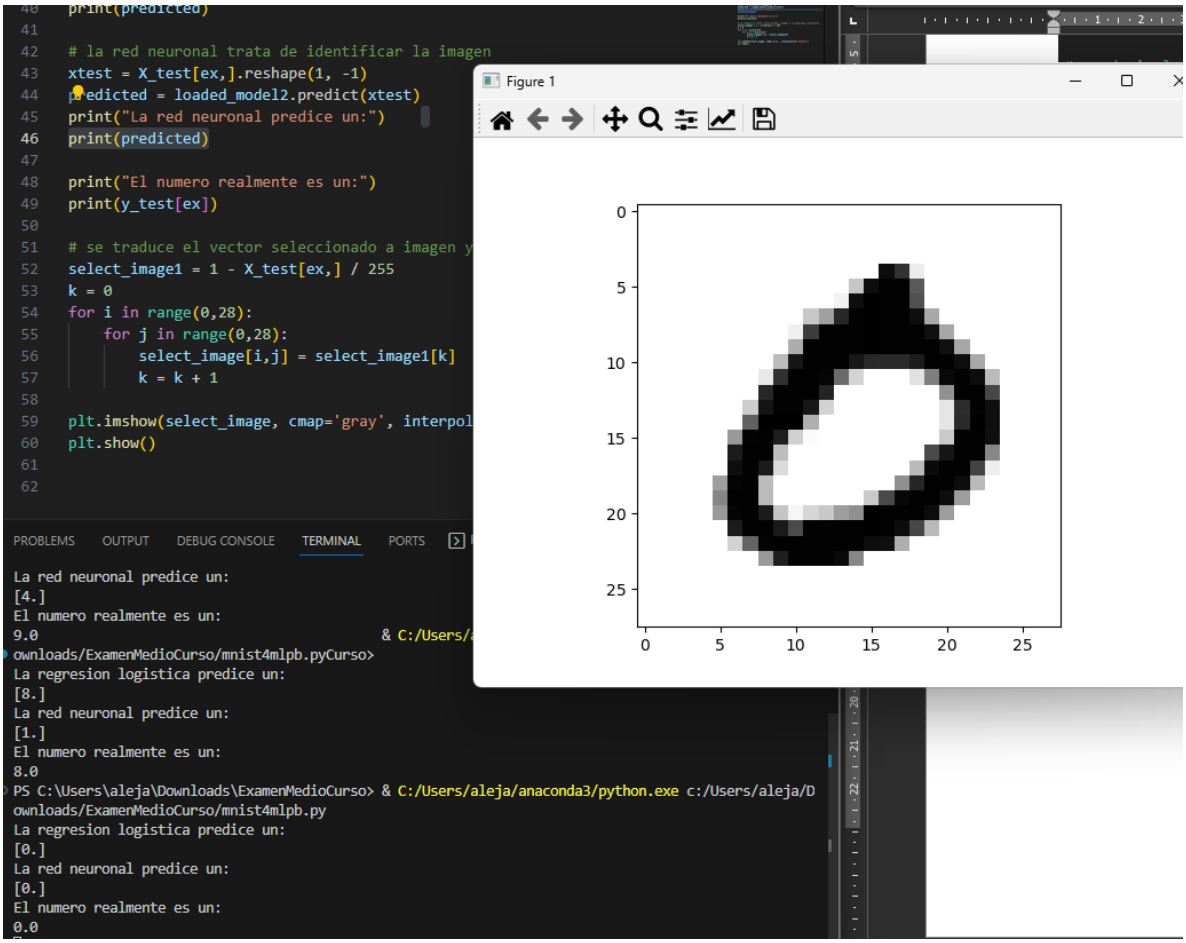
Guardamos este nuevo modelo en un archivo llamado “finalized model”.

```
# Guardar el modelo de regresión logística en un archivo .sav
logistic_reg_filename = 'finalized_model.sav'
pickle.dump(logistic_reg_model, open(logistic_reg_filename, 'wb'))
```

Corremos el programa mnist2b.py, nos va generar los 2 archivos necesarios: finalized_model.sav y finalized_model2.sav

Resultados

Finalmente corremos el programa mnist4mlpb.py y obtenemos los resultados:



Conclusiones:

Ambos modelos tienen diferentes resultados, a mi parecer la regresión logística es más acertada con los resultados. No sería mala idea añadir mas información a imprimir ciertas cosas. Utilizar el mismo conjunto de datos puede generar modelos con precisiones diferentes.