

**UNIVERSIDAD PANAMERICANA
CAMPUS MIXCOAC**

Inteligencia Artificial

Informe
Proyecto Tercer Parcial

Alumnos:
Leonardo Kenji Minemura
José Alexander Romero Mitiaevas
Angel Esqueda Ochoa

Fecha de entrega:
31/05/23

Índice

<u>Índice</u>	<u>2</u>
<u>1. Ejecución del Código</u>	<u>3</u>
<u>2. Definición del Problema</u>	<u>17</u>
<u>3. Conjunto de Datos</u>	<u>18</u>
<u>4. Clasificación</u>	<u>18</u>
a. Red neuronal base	<u>18</u>
b. Optimización de hiper-parámetros	<u>21</u>
c. Red neuronal con mejor desempeño	<u>23</u>
d. Métricas de evaluación	<u>23</u>
e. Análisis de los mejores resultados	<u>25</u>
<u>5. Regresión</u>	<u>25</u>
a. Red neuronal base	<u>25</u>
b. Optimización de hiper-parámetros	<u>28</u>
c. Red neuronal con mejor desempeño	<u>28</u>
d. Métricas de evaluación	<u>28</u>
e. Análisis de los mejores resultados	<u>30</u>
<u>6. Conclusión</u>	<u>30</u>

1. Ejecución del Código

Importación de librerías

```
#Dependencias

##Load necessary libraries
# libraries to manipulate the data and to visualise it
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 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 classification
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
# the evaluation metrics for regression
from sklearn.metrics import r2_score
from sklearn.metrics import max_error
from sklearn.metrics import explained_variance_score
from math import sqrt
import seaborn as sn
# for hyper parameter tuning
from sklearn.model_selection import GridSearchCV
```

Muestra del dataset - En este punto los datos existen en múltiples formatos

```
##carga un archivo CSV en un DataFrame, muestra información sobre el tamaño del conjunto de datos y muestra una vista previa de las primeras filas.
df = pd.read_csv(filepath_or_buffer = '/content/adult.data', names = ['age','work class','fnlwgt','education', 'education num', 'marital status','occupation'
data = [df]
print('The dataset contains {} observations and {} features\n'.format(df.shape[0],df.shape[1]))

df.head()

The dataset contains 32561 observations and 15 features

   age   work class fnlwgt education  education num marital status occupation relationship race sex capital gain capital loss hours per week native country salary
0   39  State-gov    77516  Bachelors        13 Never-married      Adm-clerical Not-in-family  White  Male     2174         0          40 United-States <=50K
1   50  Self-emp-not-inc    83311  Bachelors        13 Married-civ-spouse Exec-managerial Husband  White  Male      0         0          13 United-States <=50K
2   38       Private    215646  HS-grad          9 Divorced      Handlers-cleaners Not-in-family  White  Male      0         0          40 United-States <=50K
3   53       Private    234721      11th          7 Married-civ-spouse      Handlers-cleaners Husband  Black  Male      0         0          40 United-States <=50K
4   28       Private    338409  Bachelors        13 Married-civ-spouse Prof-specialty        Wife  Black Female      0         0          40      Cuba <=50K
```

Comenzamos analizando, visualizando y limpiando los datos del dataset.

Checar datos a evaluar

```
# checar el numero de observaciones para 'salary'
df.groupby('salary').size()

salary
<=50K      24720
>50K       7841
dtype: int64
```

Análisis de 5 números para analizar estadísticamente los datos

```
# Análisis de 5 números para analizar estadísticamente los datos
df.describe()
```

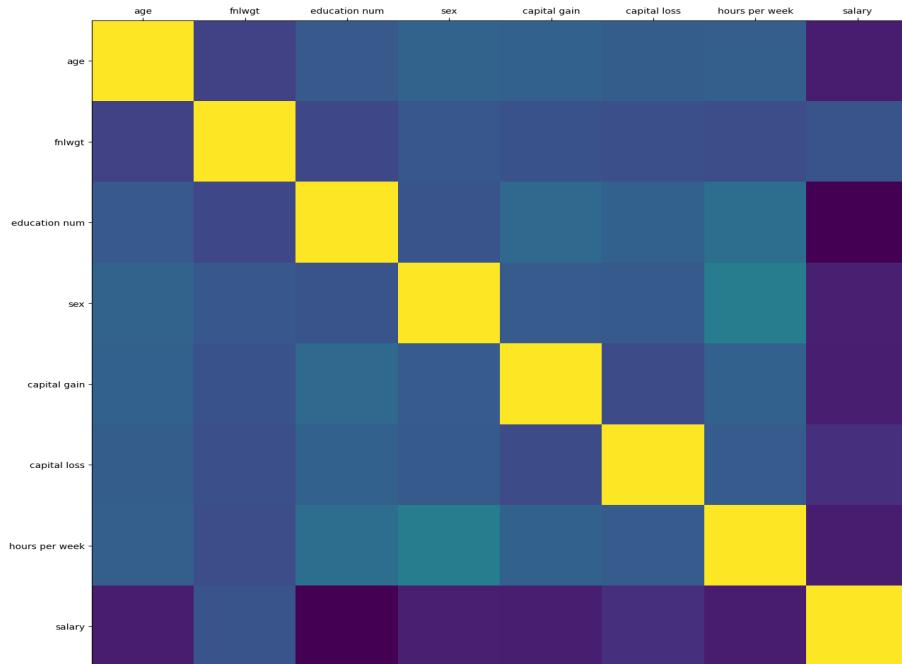
	age	fnlwgt	education num	capital gain	capital loss	hours per week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Conversión de datos: Conversión de string a int para 'salary', 'sex' y 'native country' - La conversión es necesaria para realizar la regresión.

● #Convertir 'salario' a int salary_map={'<50K':1,'>50K':0} df['salary']=df['salary'].map(salary_map).astype(int) df.head(4)																																																																											
<table border="1"> <thead> <tr> <th>age</th><th>work class</th><th>fnlwgt</th><th>education</th><th>education num</th><th>marital status</th><th>occupation</th><th>relationship</th><th>race</th><th>sex</th><th>capital gain</th><th>capital loss</th><th>hours per week</th><th>native country</th><th>salary</th></tr> </thead> <tbody> <tr> <td>0 39</td><td>State-gov</td><td>77516</td><td>Bachelors</td><td>13</td><td>Never-married</td><td>Adm-clerical</td><td>Not-in-family</td><td>White</td><td>Male</td><td>2174</td><td>0</td><td>40</td><td>United-States</td><td>1</td></tr> <tr> <td>1 50</td><td>Self-emp-not-inc</td><td>83311</td><td>Bachelors</td><td>13</td><td>Married-civ-spouse</td><td>Exec-managerial</td><td>Husband</td><td>White</td><td>Male</td><td>0</td><td>0</td><td>13</td><td>United-States</td><td>1</td></tr> <tr> <td>2 38</td><td>Private</td><td>215646</td><td>HS-grad</td><td>9</td><td>Divorced</td><td>Handlers-cleaners</td><td>Not-in-family</td><td>White</td><td>Male</td><td>0</td><td>0</td><td>40</td><td>United-States</td><td>1</td></tr> <tr> <td>3 53</td><td>Private</td><td>234721</td><td>11th</td><td>7</td><td>Married-civ-spouse</td><td>Handlers-cleaners</td><td>Husband</td><td>Black</td><td>Male</td><td>0</td><td>0</td><td>40</td><td>United-States</td><td>1</td></tr> </tbody> </table>	age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	0 39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	1	1 50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	1	2 38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	1	3 53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	1
age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary																																																													
0 39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	1																																																													
1 50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	1																																																													
2 38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	1																																																													
3 53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	1																																																													
● #Convertir 'sex' a int df['sex'] = df['sex'].map({'Male':1,'Female':0}).astype(int) df.head(4)																																																																											
<table border="1"> <thead> <tr> <th>age</th><th>work class</th><th>fnlwgt</th><th>education</th><th>education num</th><th>marital status</th><th>occupation</th><th>relationship</th><th>race</th><th>sex</th><th>capital gain</th><th>capital loss</th><th>hours per week</th><th>native country</th><th>salary</th></tr> </thead> <tbody> <tr> <td>0 39</td><td>State-gov</td><td>77516</td><td>Bachelors</td><td>13</td><td>Never-married</td><td>Adm-clerical</td><td>Not-in-family</td><td>White</td><td>1</td><td>2174</td><td>0</td><td>40</td><td>United-States</td><td>1</td></tr> <tr> <td>1 50</td><td>Self-emp-not-inc</td><td>83311</td><td>Bachelors</td><td>13</td><td>Married-civ-spouse</td><td>Exec-managerial</td><td>Husband</td><td>White</td><td>1</td><td>0</td><td>0</td><td>13</td><td>United-States</td><td>1</td></tr> <tr> <td>2 38</td><td>Private</td><td>215646</td><td>HS-grad</td><td>9</td><td>Divorced</td><td>Handlers-cleaners</td><td>Not-in-family</td><td>White</td><td>1</td><td>0</td><td>0</td><td>40</td><td>United-States</td><td>1</td></tr> <tr> <td>3 53</td><td>Private</td><td>234721</td><td>11th</td><td>7</td><td>Married-civ-spouse</td><td>Handlers-cleaners</td><td>Husband</td><td>Black</td><td>1</td><td>0</td><td>0</td><td>40</td><td>United-States</td><td>1</td></tr> </tbody> </table>	age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	0 39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	1	2174	0	40	United-States	1	1 50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	1	0	0	13	United-States	1	2 38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	1	0	0	40	United-States	1	3 53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	1	0	0	40	United-States	1
age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary																																																													
0 39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	1	2174	0	40	United-States	1																																																													
1 50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	1	0	0	13	United-States	1																																																													
2 38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	1	0	0	40	United-States	1																																																													
3 53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	1	0	0	40	United-States	1																																																													
● # Convertir 'native country' a int for dataset in data: dataset.loc[dataset['native country'] != ' United-States', 'native country'] = 'Non-US' dataset.loc[dataset['native country'] == ' United-States', 'native country'] = 'US' df['native country'] = df['native country'].map({'US':1,'Non-US':0}).astype(int) df.head(4)																																																																											
<table border="1"> <thead> <tr> <th>age</th><th>work class</th><th>fnlwgt</th><th>education</th><th>education num</th><th>marital status</th><th>occupation</th><th>relationship</th><th>race</th><th>sex</th><th>capital gain</th><th>capital loss</th><th>hours per week</th><th>native country</th><th>salary</th></tr> </thead> <tbody> <tr> <td>0 39</td><td>State-gov</td><td>77516</td><td>Bachelors</td><td>13</td><td>Never-married</td><td>Adm-clerical</td><td>Not-in-family</td><td>White</td><td>1</td><td>2174</td><td>0</td><td>40</td><td>1</td><td>1</td></tr> <tr> <td>1 50</td><td>Self-emp-not-inc</td><td>83311</td><td>Bachelors</td><td>13</td><td>Married-civ-spouse</td><td>Exec-managerial</td><td>Husband</td><td>White</td><td>1</td><td>0</td><td>0</td><td>13</td><td>1</td><td>1</td></tr> <tr> <td>2 38</td><td>Private</td><td>215646</td><td>HS-grad</td><td>9</td><td>Divorced</td><td>Handlers-cleaners</td><td>Not-in-family</td><td>White</td><td>1</td><td>0</td><td>0</td><td>40</td><td>1</td><td>1</td></tr> <tr> <td>3 53</td><td>Private</td><td>234721</td><td>11th</td><td>7</td><td>Married-civ-spouse</td><td>Handlers-cleaners</td><td>Husband</td><td>Black</td><td>1</td><td>0</td><td>0</td><td>40</td><td>1</td><td>1</td></tr> </tbody> </table>	age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	0 39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	1	2174	0	40	1	1	1 50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	1	0	0	13	1	1	2 38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	1	0	0	40	1	1	3 53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	1	0	0	40	1	1
age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary																																																													
0 39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	1	2174	0	40	1	1																																																													
1 50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	1	0	0	13	1	1																																																													
2 38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	1	0	0	40	1	1																																																													
3 53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	1	0	0	40	1	1																																																													

Gráfica de correlación: para analizar cómo se relacionan los datos y cuáles son importantes.

```
# Crear una matriz de correlación para analizar como se relacionan los datos
corr= df.corr()
fig, ax = plt.subplots(figsize=(15,15))
ax.matshow(corr)
plt.xticks(range(len(corr.columns)),corr.columns)
plt.yticks(range(len(corr.columns)),corr.columns)
plt.show()
```



Colores oscuros - correlación negativa

Colores claros - correlación positiva

Deshacerse de valores innecesarios

```
[8] # Deshacerse de los valores '?'
df['native country'] = df['native country'].replace(' ?',np.nan)
df['work class'] = df['work class'].replace(' ?',np.nan)
df['occupation'] = df['occupation'].replace(' ?',np.nan)

df.dropna(how='any',inplace=True)
```

Gráfica de línea : 'hours per week' y 'salary'(al realizar la comparación 'salary se toma como media en todas las gráficas')

```

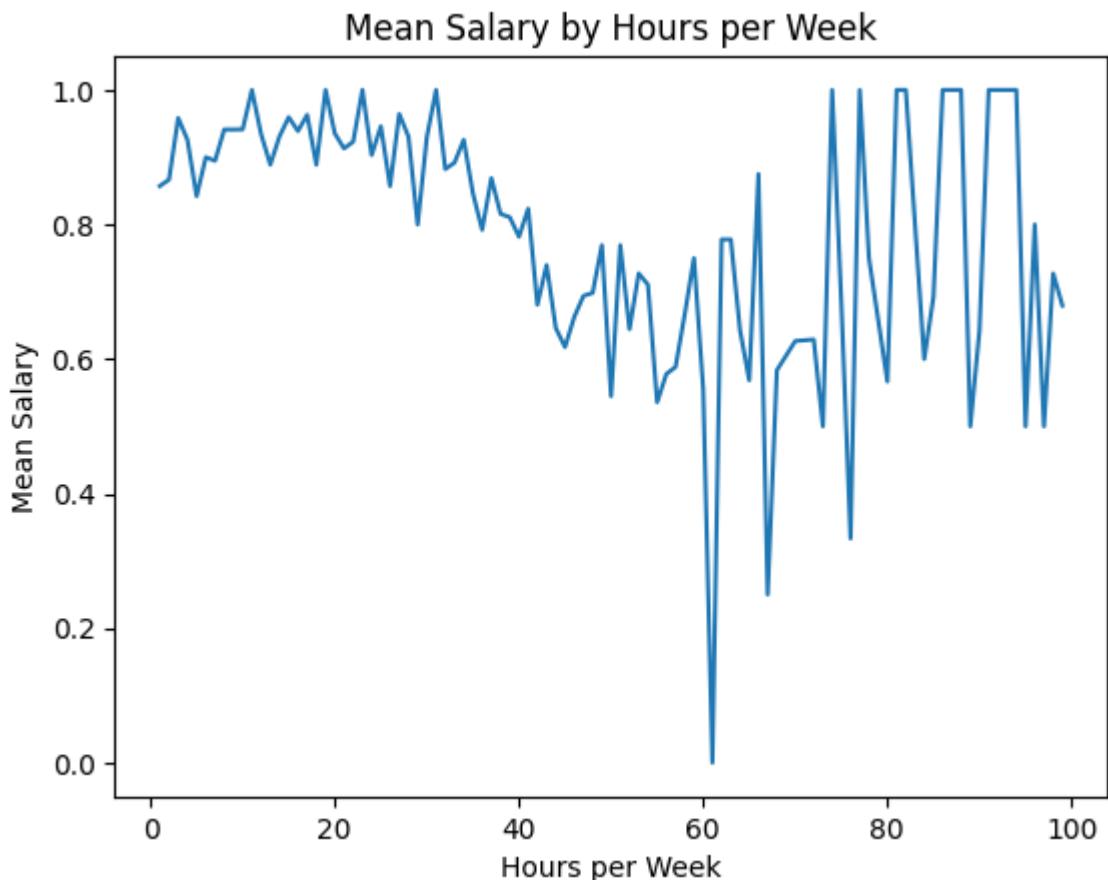
# Graficar la relacion entre 'hours per week' y 'salary'
mean_salary_by_hours = df.groupby('hours per week')['salary'].mean()

plt.plot(mean_salary_by_hours.index, mean_salary_by_hours.values)
plt.xlabel('Hours per Week')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by Hours per Week')

plt.show()

```

Gráfica obtenida - Se puede observar que alrededor del punto medio la media salarial baja.



Gráfica de barra: relación entre 'relationship' y 'salary'

```

# Graficar la relacion entre 'relationship' y 'salary'
mean_salary_by_relationship = df.groupby('relationship')['salary'].mean().reset_index()

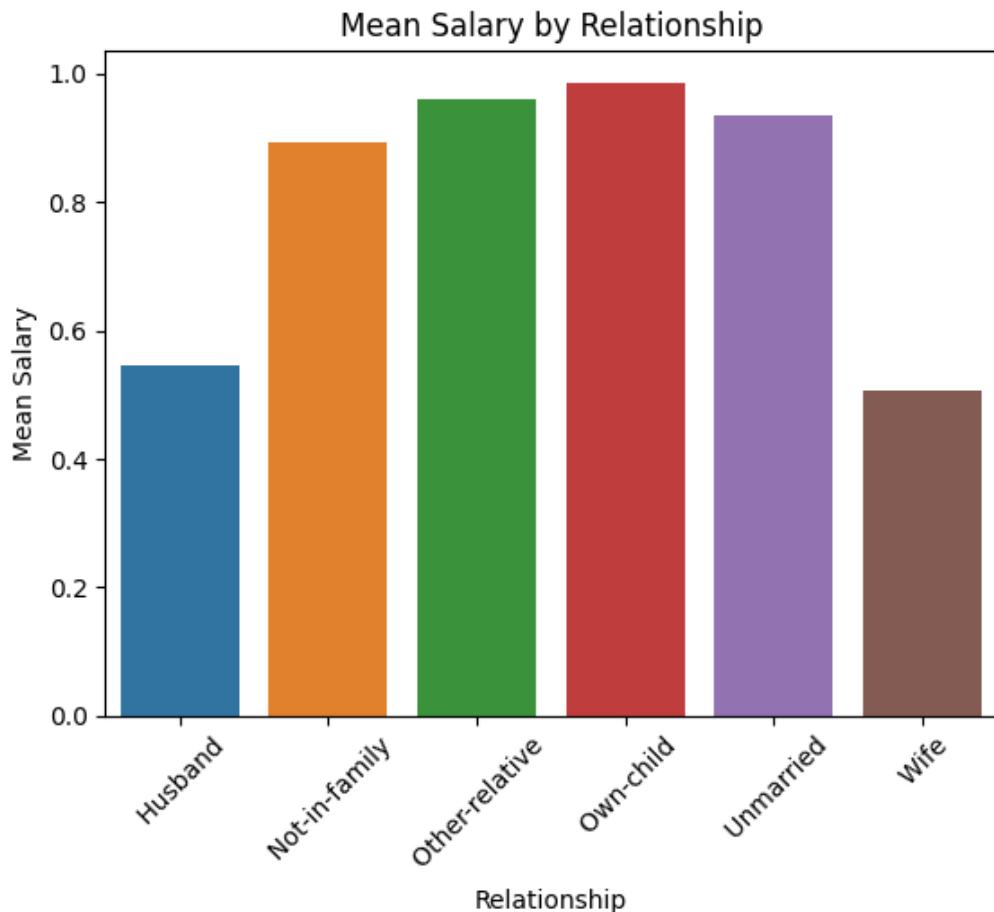
sns.barplot(x='relationship', y='salary', data=mean_salary_by_relationship)
plt.xlabel('Relationship')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by Relationship')

plt.xticks(rotation=45)

plt.show()

```

Gráfica obtenida - Se puede observar que hay una diferencia entre la gente en una relación y la gente que no



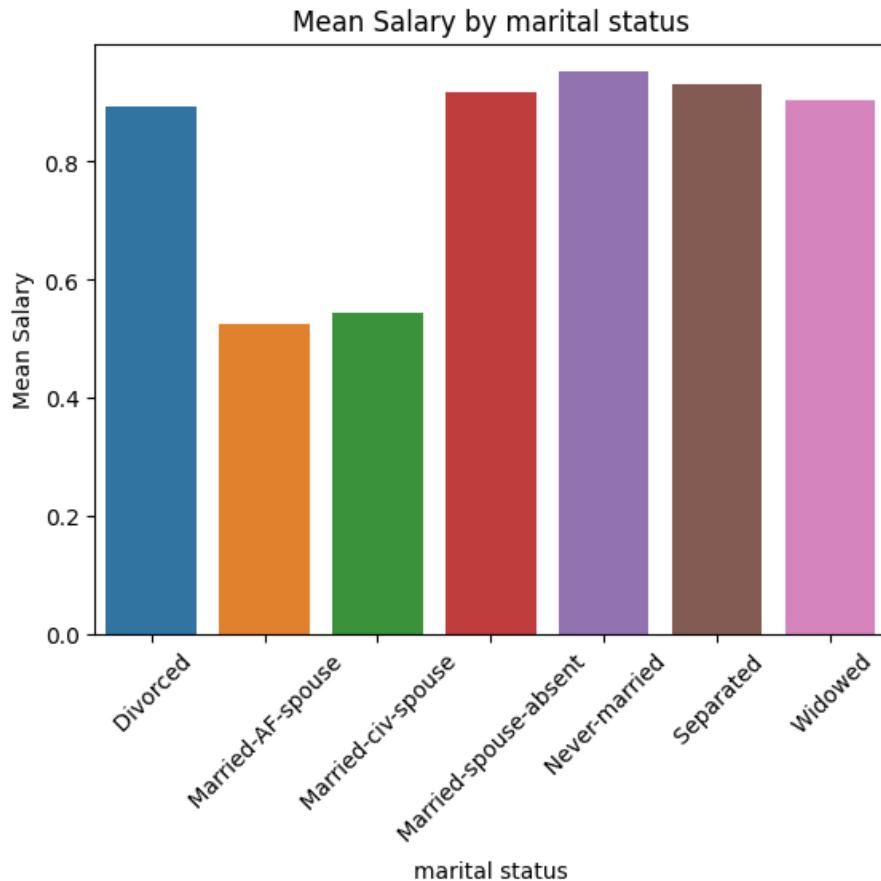
Gráfica de barra: relación entre 'marital status' y 'salary'

```
## Graficar la relacion entre 'marital status' y 'salary'
mean_salary_by_marital_status = df.groupby('marital status')['salary'].mean().reset_index()

sns.barplot(x='marital status', y='salary', data=mean_salary_by_marital_status)
plt.xlabel('marital status')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by marital status')

plt.xticks(rotation=45)
|
plt.show()
```

Gráfica obtenida - se puede observar la diferencia entre gente que está con una pareja y no.



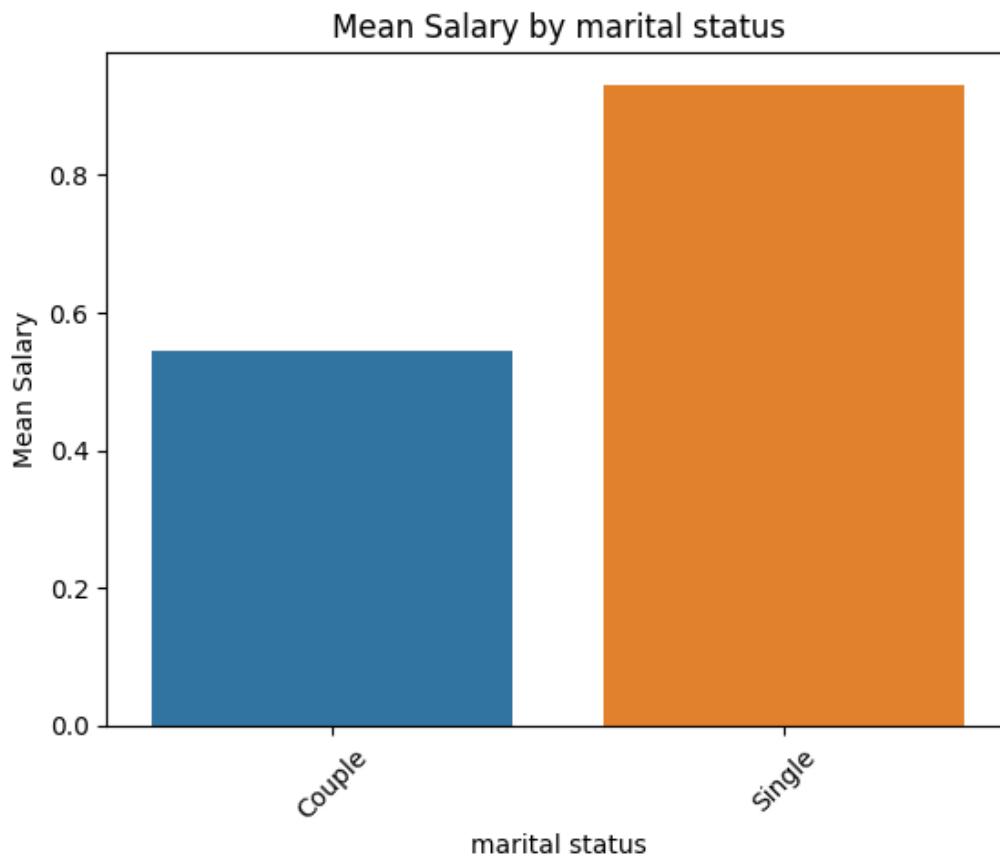
Conversión de datos: Agrupar datos a 'Single' y 'Couple' ya que es la diferencia que nos importa.

```
# Agrupar los datos en 'Single' y 'Couple'
df['marital status'] = df['marital status'].replace(['Divorced', 'Married-spouse-absent', 'Never-married', 'Separated', 'Widowed'], 'Single')
df['marital status'] = df['marital status'].replace(['Married-AF-spouse', 'Married-civ-spouse'], 'Couple')

df.head(4)
```

	age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary
0	39	State-gov	77516	Bachelors	13	Single	Adm-clerical	Not-in-family	White	1	2174	0	40	1	1
1	50	Self-emp-not-inc	83311	Bachelors	13	Couple	Exec-managerial	Husband	White	1	0	0	13	1	1
2	38	Private	215646	HS-grad	9	Single	Handlers-cleaners	Not-in-family	White	1	0	0	40	1	1
3	53	Private	234721	11th	7	Couple	Handlers-cleaners	Husband	Black	1	0	0	40	1	1

Gráfica de barra: relación entre 'marital status' (Después de agrupar) y 'salary'



Grafica mapa de calor: Relación ‘salario’ por ‘relationship’ y ‘marital status’

```
# Graficar la relacion entre 'relatioship', 'marital status' y 'salary'
mean_salary_by_relationship_marital = df.groupby(['relationship', 'marital status'])['salary'].mean().reset_index()

pivot_table = mean_salary_by_relationship_marital.pivot(index='relationship', columns='marital status', values='salary')

sns.heatmap(pivot_table, cmap='YlGnBu', annot=True, fmt='.2f')

plt.xlabel('Marital Status')
plt.ylabel('Relationship')
plt.title('Mean Salary by Relationship and Marital Status')

plt.show()
```

Grafica obtenida - Se puede observar que claramente existe una diferencia entre la gente con pareja y no.



Conversión de datos: 'marital status' a int

```
# Convertir 'marital status' a int
df['marital status'] = df['marital status'].map({'Couple':0,'Single':1})

rel_map = {' Unmarried':0,' Wife':1,' Husband':2,' Not-in-family':3,' Own-child':4,' Other-relative':5}

df['relationship'] = df['relationship'].map(rel_map)

df.head(4)
```

	age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary
0	39	State-gov	77516	Bachelors	13	1	Adm-clerical	3	White	1	2174	0	40	1	1
1	50	Self-emp-not-inc	83311	Bachelors	13	0	Exec-managerial	2	White	1	0	0	13	1	1
2	38	Private	215646	HS-grad	9	1	Handlers-cleaners	3	White	1	0	0	40	1	1
3	53	Private	234721	11th	7	0	Handlers-cleaners	2	Black	1	0	0	40	1	1

Gráfica de barra: relación entre 'race' y 'salary'

```
# Graficar la relacion entre 'race' y 'salary'
mean_salary_by_race = df.groupby('race')[['salary']].mean().reset_index()

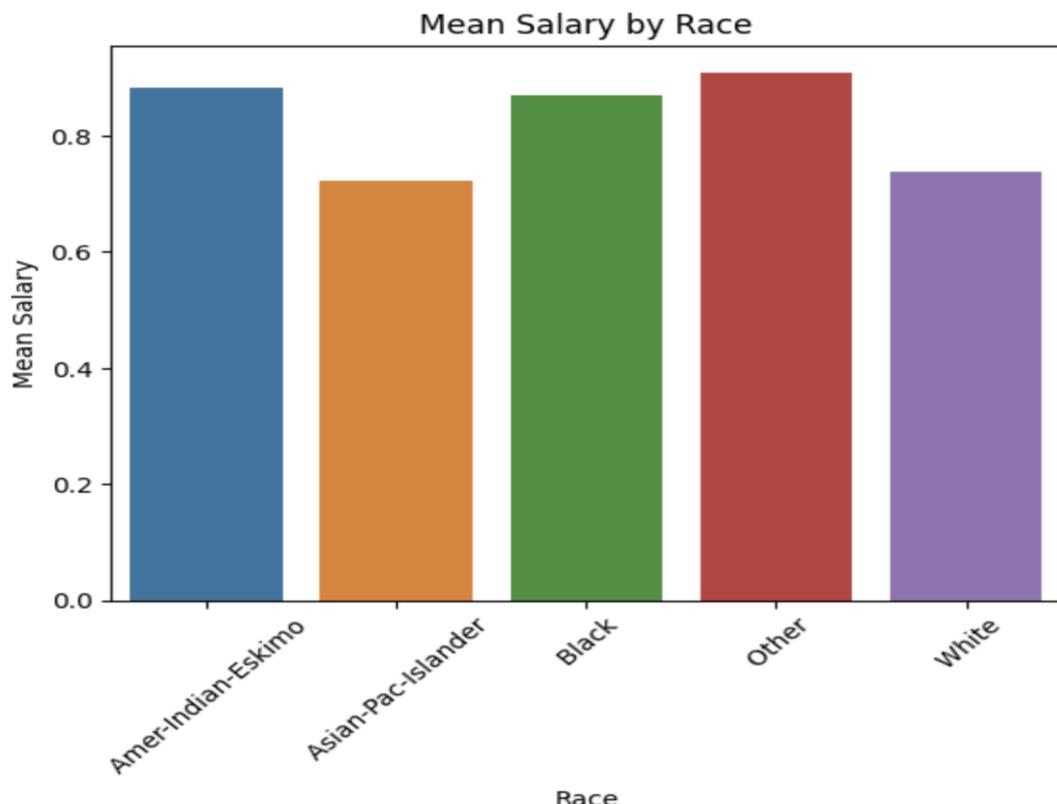
sns.barplot(x='race', y='salary', data=mean_salary_by_race)

plt.xlabel('Race')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by Race')

plt.xticks(rotation=45)

plt.show()
```

Gráfica obtenida - Se puede observar que la raza y el salario si tienen una relación



Conversión de datos: 'race' a int

```
#Convertir 'race' a int
race_map={' White':0,' Amer-Indian-Eskimo':1,' Asian-Pac-Islander':2,' Black':3,' Other':4}
df['race']= df['race'].map(race_map)

df.head(4)
```

	age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	
0	39	State-gov	77516	Bachelors	13	1	Adm-clerical		3	0	1	2174	0	40	1	1
1	50	Self-emp-not-inc	83311	Bachelors	13	0	Exec-managerial		2	0	1	0	0	13	1	1
2	38	Private	215646	HS-grad	9	1	Handlers-cleaners		3	0	1	0	0	40	1	1
3	53	Private	234721	11th	7	0	Handlers-cleaners		2	3	1	0	0	40	1	1

Grafica de barra: relación entre 'occupation' y 'salary'

```
# Graficar la relacion entre 'occupation' y 'salary'
mean_salary_by_occupation = df.groupby('occupation')['salary'].mean().reset_index()

sn.barplot(x='occupation', y='salary', data=mean_salary_by_occupation)
plt.xlabel('Occupation')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by Occupation')

plt.xticks(rotation=45)

plt.show()
```

Gráfica obtenida: Se observa que 'occupation' si afecta a 'salary'. Sin embargo, independientemente de la ocupación, 'work class' afecta mayormente.



Gráfica de barra: relación entre 'work class' y 'salary'

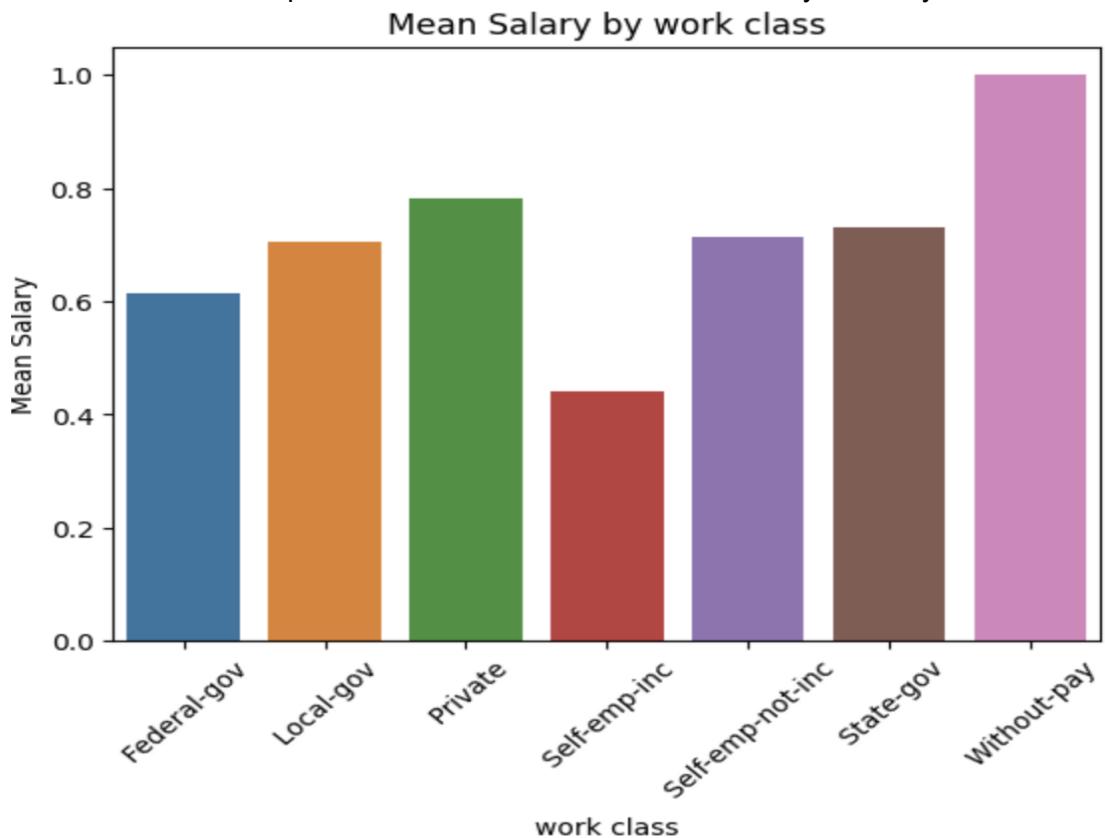
```
# Graficar la relacion entre 'work class' y 'salary'
mean_salary_by_work_class = df.groupby('work class')['salary'].mean().reset_index()

sns.barplot(x='work class', y='salary', data=mean_salary_by_work_class)
plt.xlabel('work class')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by work class')

plt.xticks(rotation=45)

plt.show()
```

Gráfica obtenida - se puede observar una relación más clara y con mayor diferencia.



Conversión de datos: Agrupar datos de 'work class' y meterlos a una nueva columna llamada 'employment type'

```
# Agrupar los datos de 'work class' y meterlos a una nueva columna llamada 'employment type'
def f(x):
    if x['work class'] == ' Federal-gov' or x['work class']==' Local-gov' or x['work class']==' State-gov': return 'govt'
    elif x['work class'] == ' Private':return 'private'
    elif x['work class'] == ' Self-emp-inc' or x['work class'] == ' Self-emp-not-inc': return 'self_employed'
    else: return 'without_pay'

df['employment type']=df.apply(f, axis=1)
df.head(4)
```

	age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	employment type
0	39	State-gov	77516	Bachelors	13	1	Adm-clerical	3	0	1	2174	0	40	1	1	govt
1	50	Self-emp-not-inc	83311	Bachelors	13	0	Exec-managerial	2	0	1	0	0	13	1	1	self-employed
2	38	Private	215646	HS-grad	9	1	Handlers-cleaners	3	0	1	0	0	40	1	1	private
3	53	Private	234721	11th	7	0	Handlers-cleaners	2	3	1	0	0	40	1	1	private

Grafica de barra: Relación entre 'employment type' y 'salary'

```
# Graficar la relacion entre 'employment type' y 'salary'
mean_salary_by_workclass = df.groupby('employment type')['salary'].mean().reset_index()

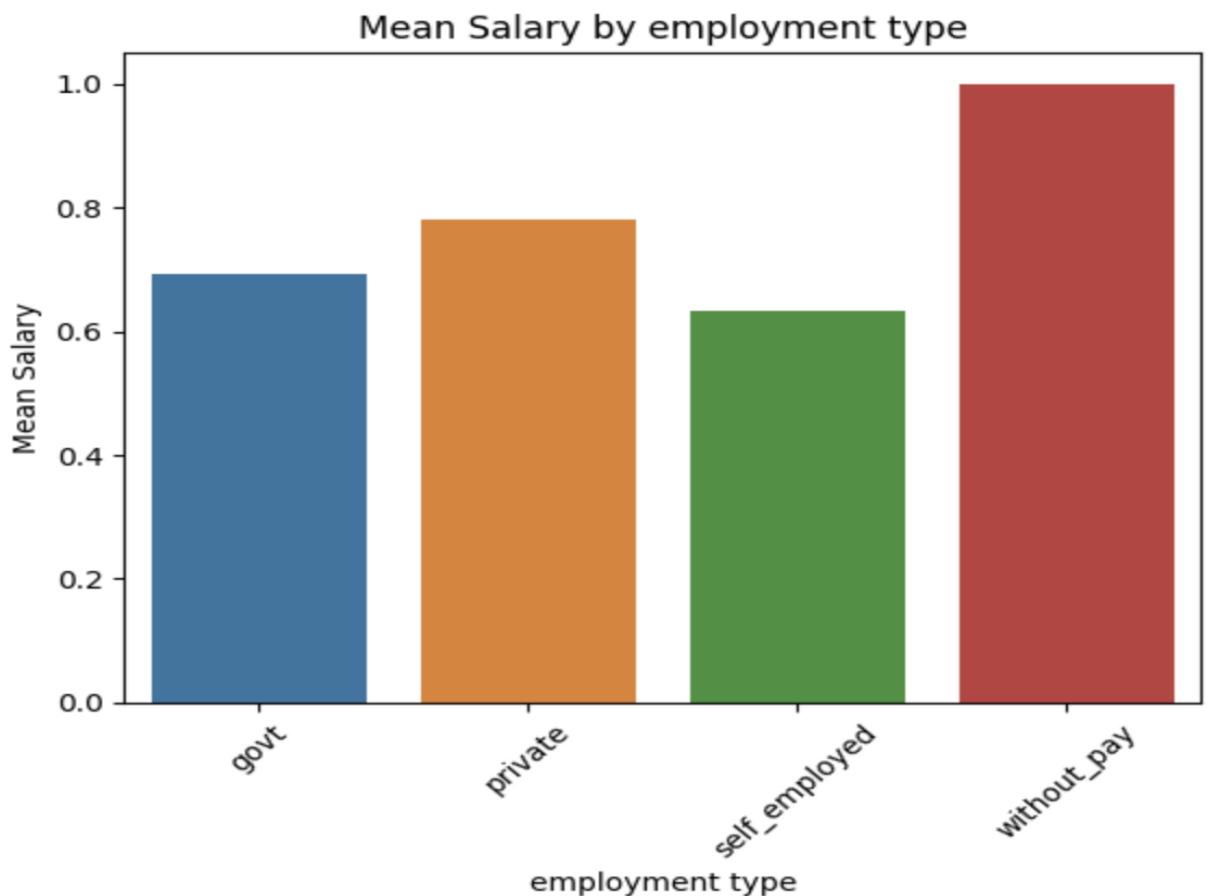
sns.barplot(x='employment type', y='salary', data=mean_salary_by_workclass)

plt.xlabel('employment type')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by employment type')

plt.xticks(rotation=45)

plt.show()
```

Gráfica obtenida - Se han simplificado los datos y se pueden observar las diferencias con mayor facilidad



Conversión de datos: 'employment type' a int

```
#Convertir 'employment type' a int
employmenttype_map = {'govt':0,'private':1,'self-employed':2,'without_pay':3}
df['employment type'] = df['employment type'].map(employmenttype_map)
df.head(4)
```

age	work class	fnlwgt	education	education num	marital status	occupation	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	employment type	
0	39	State-gov	77516	Bachelors	13	1	Adm-clerical	3	0	1	2174	0	40	1	1	0
1	50	Self-emp-not-inc	83311	Bachelors	13	0	Exec-managerial	2	0	1	0	0	13	1	1	2
2	38	Private	215646	HS-grad	9	1	Handlers-cleaners	3	0	1	0	0	40	1	1	1
3	53	Private	234721	11th	7	0	Handlers-cleaners	2	3	1	0	0	40	1	1	1

Conversión de datos: Eliminar datos innecesarios.

```
#Eliminar datos innecesarios
df.drop(labels=['work class','education','occupation'],axis=1,inplace=True)
df.head(4)
```

	age	fnlwgt	education num	marital status	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	employment type
0	39	77516	13	1	3	0	1	2174	0	40	1	1	0
1	50	83311	13	0	2	0	1	0	0	13	1	1	2
2	38	215646	9	1	3	0	1	0	0	40	1	1	1
3	53	234721	7	0	2	3	1	0	0	40	1	1	1

Gráfica de barra: Relación entre 'education num' y 'salary'

```
# Graficar la relacion entre 'education num' y 'salary'
mean_salary_by_education_num = df.groupby('education num')['salary'].mean().reset_index()

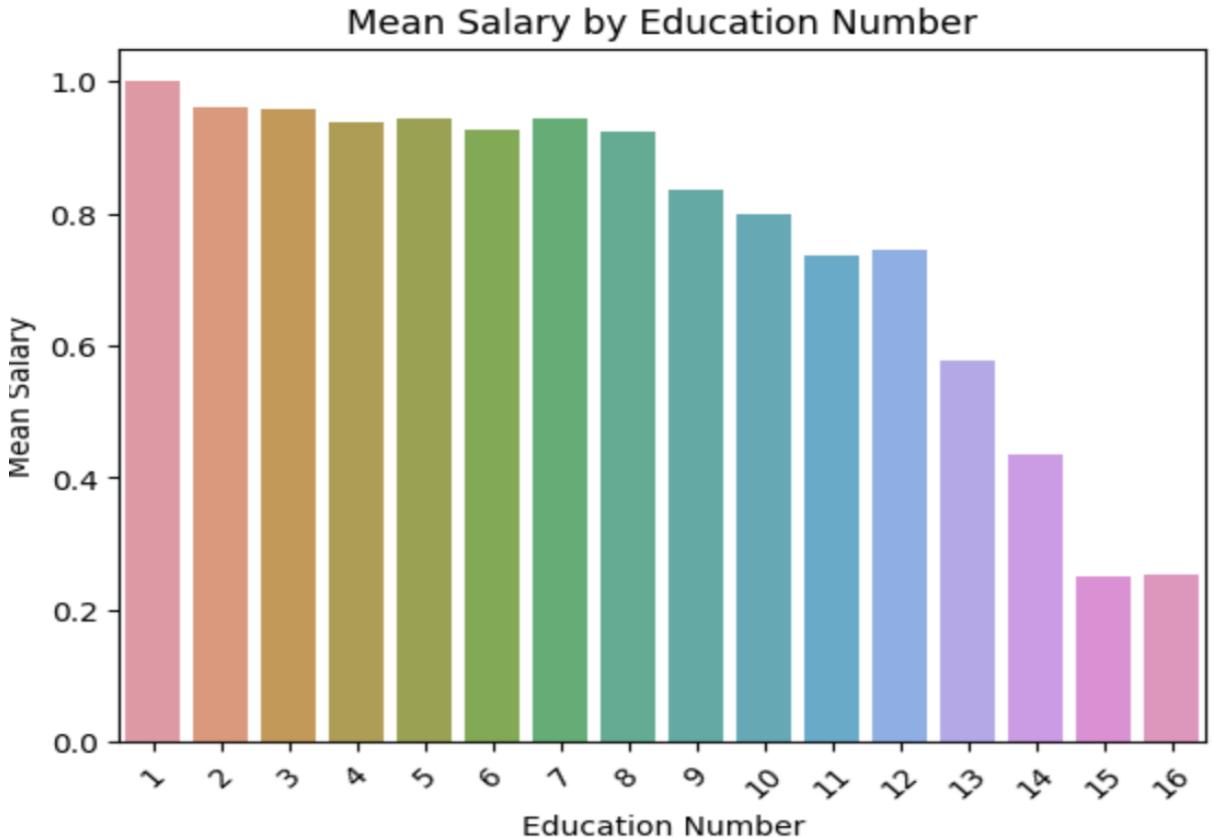
sns.barplot(x='education num', y='salary', data=mean_salary_by_education_num)

plt.xlabel('Education Number')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by Education Number')

plt.xticks(rotation=45)

plt.show()
```

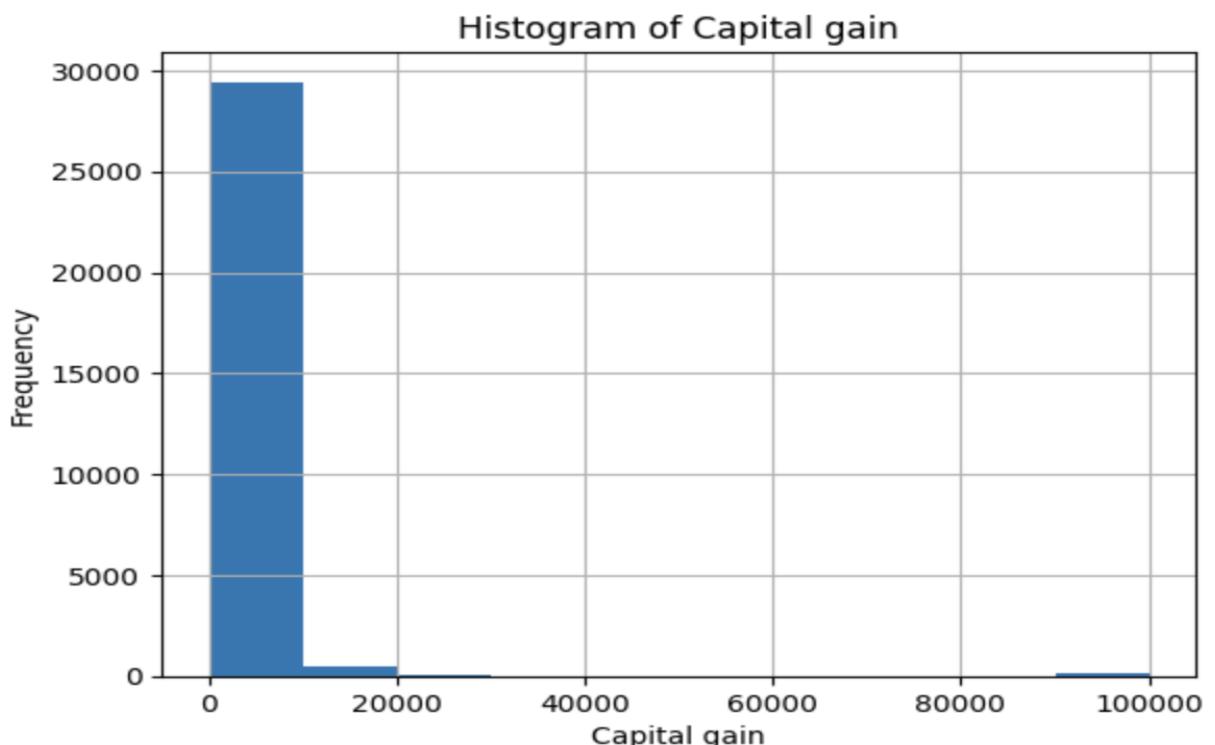
Grafica obtenida: Se puede observar una proporción inversa



Grafica histograma: Analizar el comportamiento de 'capital gain'

```
#Graficar un histograma para 'capital gain'  
df['capital gain'].hist()  
  
# Add labels and title  
plt.xlabel('Capital gain')  
plt.ylabel('Frequency')  
plt.title('Histogram of Capital gain')  
  
# Display the histogram  
plt.show()
```

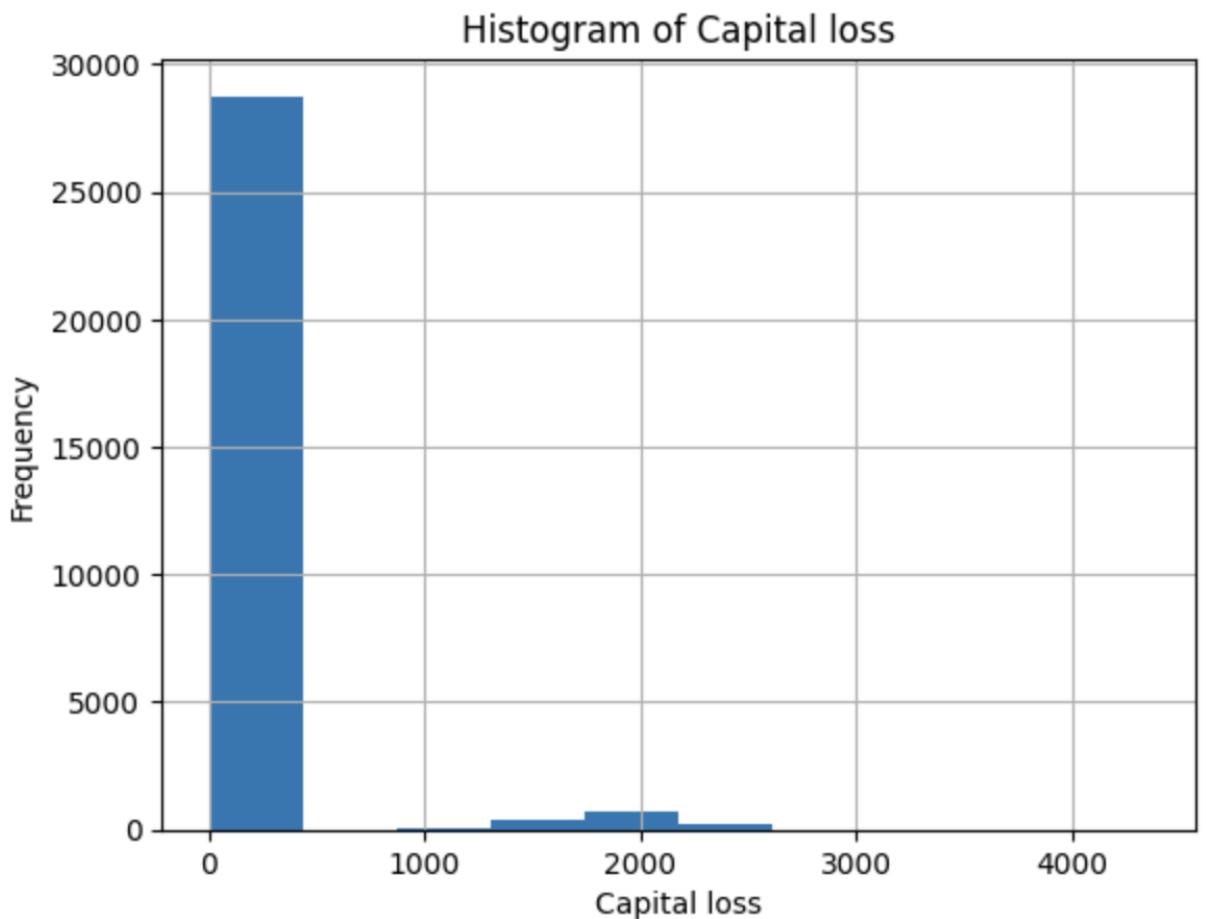
Grafica obtenida: Se puede observar que la mayoría de los valores se concentran en 0



Gráfica Histograma: Observar el comportamiento de 'capital loss'

```
#Graficar un histograma para 'capital loss'  
df['capital loss'].hist()  
  
# Add labels and title  
plt.xlabel('Capital loss')  
plt.ylabel('Frequency')  
plt.title('Histogram of Capital loss')  
  
# Display the histogram  
plt.show()
```

Grafica obtenida - se puede observar que la mayoría de los valores se concentran en 0



Conversión de datos: Se simplificará 'capital gain' y 'capital loss'. Ya que la mayoría de los valores se concentran en 0. Serán tomados como 0 o mayor a 0

```
#Simplificar los datos
df.loc[(df['capital gain'] > 0), 'capital gain'] = 1
df.loc[(df['capital gain'] == 0 , 'capital gain')]= 0
df.loc[(df['capital loss'] > 0), 'capital loss'] = 1
df.loc[(df['capital loss'] == 0 , 'capital loss')]= 0

df.head(5)
```

	age	fnlwgt	education	num	marital status	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	employment type	
0	39	77516		13	1		3	0	1	1	0	40	1	1	0
1	50	83311		13	0		2	0	1	0	0	13	1	1	2
2	38	215646		9	1		3	0	1	0	0	40	1	1	1
3	53	234721		7	0		2	3	1	0	0	40	1	1	1

A continuación se generarán un conjunto de datos para entrenar y uno para evaluar.

```
#Crecion de training set y testing set
### Drop salary
# x = dataset sin feature
x = df.drop(['salary'], axis=1)

# convertirlo a numpy array y asignar valores
x = x.to_numpy()[:, (0,1,2,3,4,5,6,7,8,9,10,11)]

# y = dataset con feature
y = df['salary']

# separar a train y test
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.5,random_state=42)

## Visualizar training set y testing set

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

#Transformacion escalar para ajustar y normalizar los datos
sc = StandardScaler()
scaler = sc.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

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

Se elimina el valor por el cual queremos clasificar o predecir.

Se convierten los valores que queremos tomar a consideración a un array de numpy.

Con la función train_test_split se generan los conjuntos de datos.

Vista al dataset

```
Min values of the dataset are:
age           17
fnlwgt      13769
education num       1
marital status      0
relationship        0
race            0
sex              0
capital gain      0
capital loss       0
hours per week     1
native country      0
salary            0
employment type     0
dtype: int64
Max values of the dataset are:
age           90
fnlwgt      1484705
education num      16
marital status      1
relationship        5
race            4
sex              1
capital gain      1
capital loss       1
hours per week    99
native country      1
salary            1
employment type     3
dtype: int64
Min values of the scaled dataset are:
-3.5783900922996916
Max values of the scaled dataset are:
12.274597232894362
```

Ya una vez teniendo esto, el conjunto de datos está listo para pasar por clasificación y regresión.

2. Definición del Problema

El problema consiste en realizar una predicción para determinar si una persona tiene un ingreso anual superior a 50 mil dólares, utilizando un conjunto de datos extraídos por Barry Becker de la base de datos del Censo de 1994. Los registros seleccionados cumplen con las siguientes condiciones: la edad de la persona es mayor a 16 años, el ingreso ajustado es mayor a 100, el peso de muestreo final es mayor a 1 y el número de horas trabajadas por semana es mayor a 0. La tarea es clasificar correctamente cada registro como "superior a 50K" o "inferior o igual a 50K" en función de los atributos mencionados.

3. Conjunto de Datos

La base de datos seleccionada para este proyecto fue “Adult Data Set” la cual fue conseguida en la página UCI machine learning repository. En la cual se evalúan datos como la Edad, Empleador, Peso de muestreo final, Nivel educativo, Años de educación, Estado civil, Ocupación, Relación familiar, Raza, Género, Ganancias de capital, Pérdidas de capital, Horas trabajadas por semana, País de origen, Ingresos. De los cuales se toman en cuenta para poder dar un resultado más certero contando con 32,561 datos.

4. Clasificación

a.

b. Red neuronal base

La red neuronal usada para el proyecto fue Multi-layer perceptron classifier. Los parámetros introducidos para la red neuronal base fueron los siguientes:

hidden layer size = 2, max iter = 300, activation = relu, solver = adam

- hidden_layer_sizes: Especifica la arquitectura de la red neuronal, es decir, el número de capas ocultas y el número de neuronas en cada capa. En este caso, se define una única capa oculta con 2 neuronas.

- max_iter: El número máximo de iteraciones (épocas) que el algoritmo de entrenamiento realizará antes de detenerse.

- activation: La función de activación que se utilizará en las neuronas. En este caso, se utiliza la función de activación ReLU (Rectified Linear Unit), que es comúnmente utilizada en redes neuronales debido a su eficiencia y capacidad para manejar problemas no lineales.

- solver: El algoritmo utilizado para optimizar los pesos de la red neuronal durante el entrenamiento. En este caso, se utiliza el algoritmo "adam", que es un optimizador basado en el descenso de gradiente estocástico y es conocido por su eficiencia y buen rendimiento en una variedad de problemas.

Estos parámetros configuran la arquitectura y el comportamiento del MLPClassifier, que es un clasificador de redes neuronales multicapa utilizado para problemas de clasificación.

```
## definir Multi-layer Perceptron classifier
mlp_clf = MLPClassifier([
    # definir las capas
    # el numero de neuronas por cada capa
    hidden_layer_sizes=(2),
    # numero maximo de iteraciones
    max_iter = 300,
    # funcion transfer/activation
    activation = 'relu',
    # optimizador de pesos
    solver = 'adam'
])
```

Ya que definimos los parametros se comenzó el entrenamiento.

```
# Entrenar el modelo
mlp_clf.fit(X_train_scaled, y_train)
```

▼ MLPClassifier
MLPClassifier(hidden_layer_sizes=2, max_iter=300)

Ya que haya terminado de entrenar, los datos se pasan a y_pred para realizar pruebas.

```
# Pasar la dataset para realizar pruebas
y_pred = mlp_clf.predict(X_test_scaled)
y_pred
```



```
array([1, 1, 1, ..., 0, 1, 1])
```

Después de haber pasado los datos se realiza la evaluación.

```

## Se evalua como fueron los resultados
# Accuracy
print('Accuracy: {:.2f}'.format(accuracy_score(y_test, y_pred)))
# precision, recall, f1-score, and support
#   f1-score no se aplica en este caso
print(classification_report(y_test, y_pred))

Accuracy: 0.83
      precision    recall  f1-score   support

          0       0.71      0.58      0.64     3794
          1       0.87      0.92      0.89    11287

   accuracy                           0.83    15081
  macro avg       0.79      0.75      0.76    15081
weighted avg       0.83      0.83      0.83    15081

```

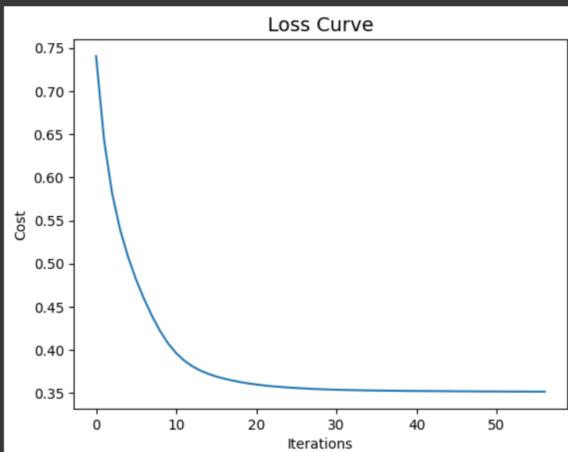
accuracy nos indica la precisión que tuvo nuestro modelo. mientras más se acerque a 1 es mejor. En este caso el f1 score no tiene ningún significado ya que solo funciona para casos binarios.

Después de esto se grafica una Loss Curve para observar cómo evolucionó nuestro modelo.

```

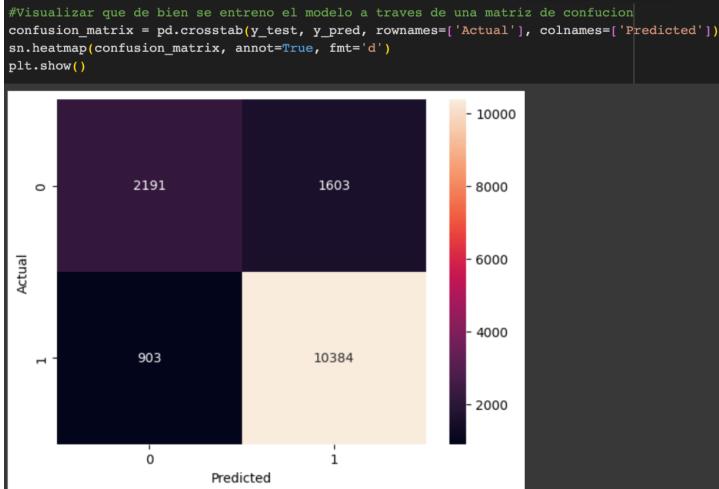
# Visualizar que de bien se entreno el modelo a traves de una Loss Curve
plt.plot(mlp_clf.loss_curve_)
plt.title("Loss Curve", fontsize=14)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.show()

```



Se puede observar como las pérdidas van disminuyendo. Por lo tanto podemos concluir que el entrenamiento fue exitoso.

Ahora graficamos un heat map para visualizar los datos obtenidos.



Aquí se puede observar como fueron las predicciones contra los valores reales. Se puede observar que ha acertado mas de lo que ha fallado.

Ahora compararemos las predicciones contra los valores actuales.

```
# observar las predicciones
y_pred_pd = pd.DataFrame(y_pred)
print('Predictions \n{}\n'.format(y_pred_pd.describe()))

# observar los datos reales
print('Groundtruth \n{}\n'.format(y_test.describe()))

Predictions
0
count    15081.000000
mean      0.794841
std       0.403831
min       0.000000
25%      1.000000
50%      1.000000
75%      1.000000
max      1.000000

Groundtruth
count    15081.000000
mean      0.748425
std       0.433933
min       0.000000
25%      0.000000
50%      1.000000
75%      1.000000
max      1.000000
Name: salary, dtype: float64
```

Se puede observar que a pesar de que existe una diferencia, los resultados obtenidos son aceptables.

c. Optimización de hiper-parámetros

Al realizar el hyper parameter tuning el camino que se escogió fue la búsqueda exhaustiva, cual es una estrategia que implica probar todas las combinaciones posibles.

```
param_grid = {
    'hidden_layer_sizes': [(150,100,50), (120,80), (100)],
    'max_iter': [50, 100, 150],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}
```

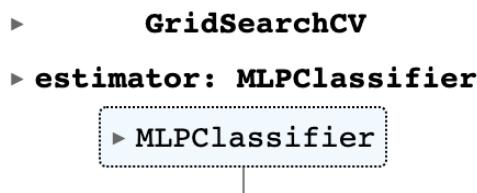
Ya que se ingresaron los parámetros se definió el enfoque de búsqueda de cuadrícula para entrenar varios modelos con validación cruzada.

```
] # definimos un enfoque de búsqueda de cuadrícula para entrenar varios modelos con validación cruzada
grid = GridSearchCV(
    # estructura del modelo en el que estamos interesados
    mlp_clf,
    # hyper parametro que queremos entrenar
    param_grid_v2,
    # usar parallelización. -1 = utiliza todos
    n_jobs = -1,
    # número de pliegues utilizados en el enfoque de validación cruzada
    cv = 5
)
```

Una vez teniendo esto, se realizó el entrenamiento.

```
# Ejecutar hyper parameter tuning
grid.fit(X_train_scaled, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_  
warnings.warn(
```



Los resultados fueron pasados de la misma manera en la red neuronal base .

Debido a que el entrenamiento no pudo ser realizado en google colab, este fue llevado a cabo en VSCode. Los parámetros con el mejor desempeño fueron los siguientes.

Dentro de google colab se encuentran los siguientes parámetros, los cuales fueron usados para poder realizar la ejecución del problema dentro de Google Colab.

```
param_grid_v2 = {
    'hidden_layer_sizes': [(150,100,50),(100)],
    'max_iter': [100],
    'activation': ['relu'],
    'solver': ['sgd'],
    'alpha': [0.0001],
    'learning_rate': ['constant'],
}
```

d. Red neuronal con mejor desempeño

La red neuronal con el mejor desempeño fue seleccionada de la siguiente manera.

```
# Conseguir el valor ganador
print('The best hyper parameter values are:\n{}'.format(grid.best_params_))
```

El valor ganador fue el siguiente

```
The best hyper parameter values are:
{'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (150, 100, 50), 'learning_rate': 'constant', 'max_iter': 100, 'solver': 'sgd'}
```

Ya que obtuvimos el valor ganador, los datos fueron convertidos en el formato que el programa pueda procesarlos y fueron ingresados para el entrenamiento del modelo.

```
#Convertir valores para que el programa los pueda reconocer
hidden_layer_sizes = grid.best_params_['hidden_layer_sizes']
hidden_layer_sizes = tuple(hidden_layer_sizes)
grid.best_params_['hidden_layer_sizes'] = hidden_layer_sizes
mlp_best = MLPClassifier(**grid.best_params_)

] #Entrenar el modelo
mlp_best.fit(X_train_scaled, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: Conva
    warnings.warn(
▼
MLPClassifier
MLPClassifier(hidden_layer_sizes=(150, 100, 50), max_iter=100, solver='sgd')
```

e. Métricas de evaluación

El código realiza la evaluación de un modelo de clasificación y muestra métricas de desempeño. Calcula y muestra la precisión (accuracy) del modelo, que representa la proporción de predicciones correctas. Luego, genera un informe de clasificación que incluye la precisión, exhaustividad (recall) y el soporte para cada clase de la variable objetivo. Estas métricas proporcionan información sobre la calidad de las predicciones del modelo en relación con las etiquetas verdaderas.

```

# Se evalua como fueron los resultados
# Accuracy
print('Accuracy: {:.2f}'.format(accuracy_score(y_test, y_pred)))
# precision, recall, f1-score, and support
#   f1-score no se aplica en este caso
print(classification_report(y_test, y_pred))
print('-----')
print(report_base)

Accuracy: 0.84
      precision    recall  f1-score   support
          0       0.72     0.59      0.65     3794
          1       0.87     0.92      0.90    11287

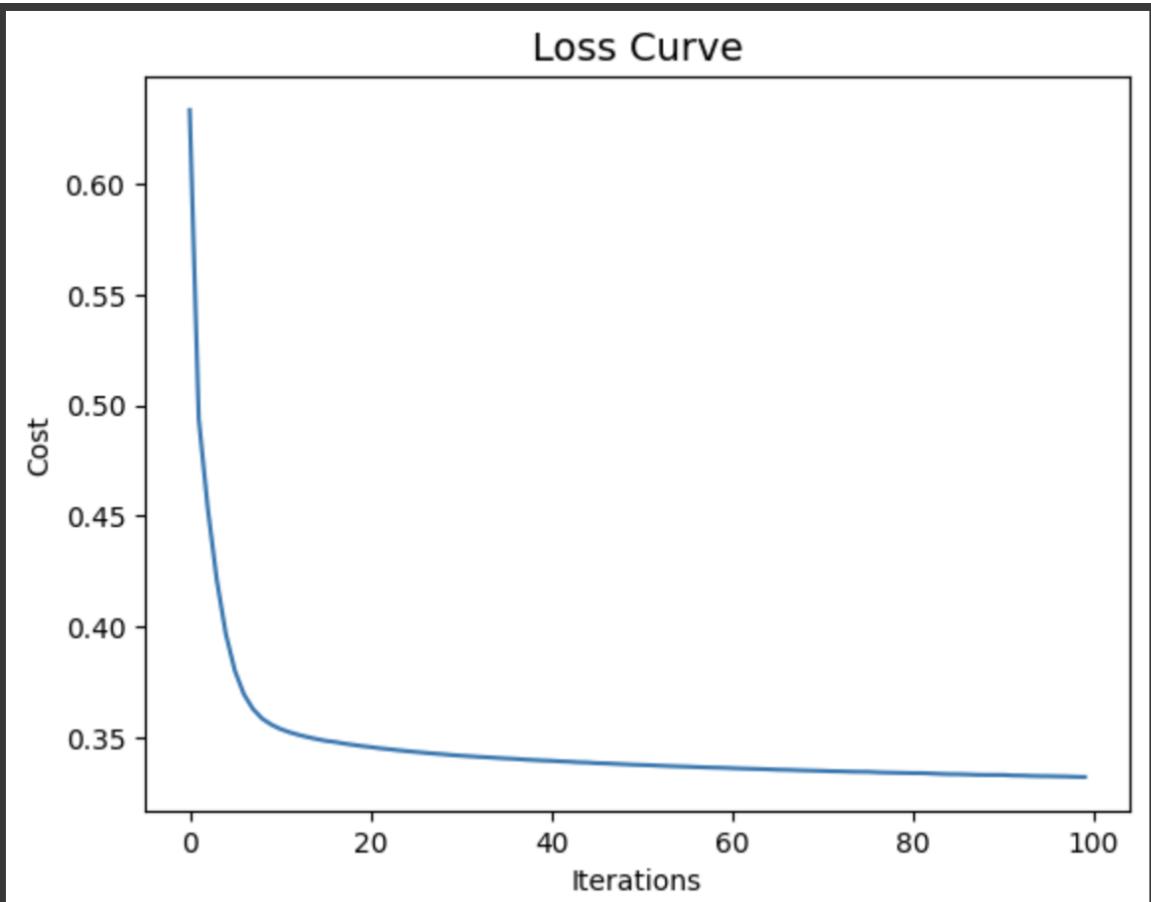
   accuracy
macro avg       0.80     0.76      0.77    15081
weighted avg     0.83     0.84      0.83    15081

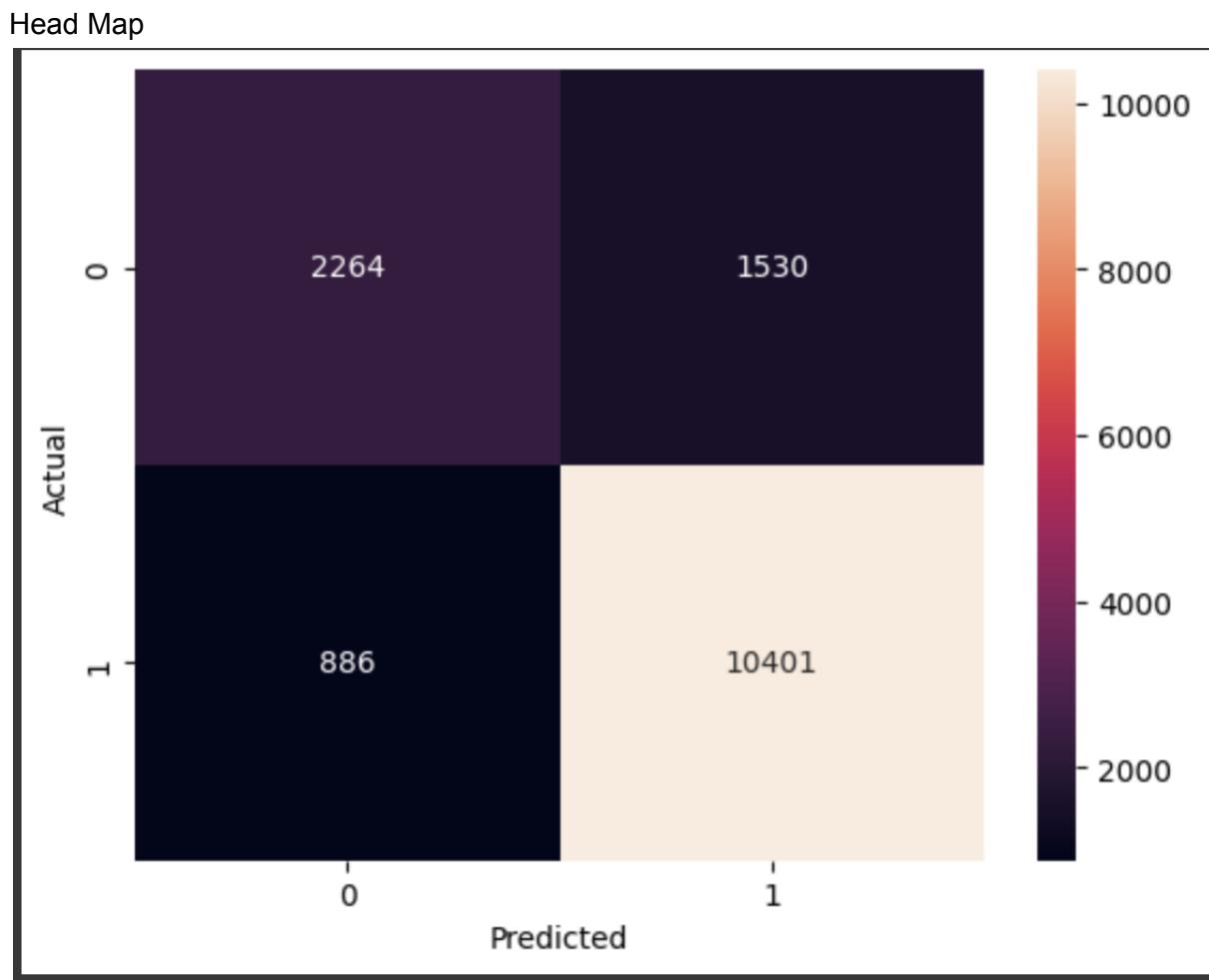
-----
      precision    recall  f1-score   support
          0       0.72     0.58      0.64     3794
          1       0.87     0.92      0.89    11287

   accuracy
macro avg       0.79     0.75      0.77    15081
weighted avg     0.83     0.84      0.83    15081

```

Loss Curve





f. Análisis de los mejores resultados

Al comparar los resultados del mejor modelo, nos podemos dar cuenta que en cuestión de accuracy no existe una gran diferencia, sin embargo al comparar las Loss curves se puede observar que el costo disminuye mucho más rápido que en el mejor modelo. Por lo tanto podemos concluir que el mejor modelo cuenta con una mayor eficiencia a pesar de que en cuestión de precisión no existe una diferencia perceptible.

5. Regresión

a. Red neuronal base

Para regresión es importante verificar que únicamente contemos con datos numéricos, ya que la regresión únicamente funcione con datos numéricos.

	#Verificar que solo existen valores numericos dataset.head()															
	age	fnlwgt	education	num	marital	status	relationship	race	sex	capital gain	capital loss	hours per week	native country	salary	employment	type
0	39	77516	13	1	3	0	1	1	1	0	0	40	1	1	0	0
1	50	83311	13	0	2	0	1	0	0	0	0	13	1	1	2	2
2	38	215646	9	1	3	0	1	0	0	0	0	40	1	1	1	1
3	53	234721	7	0	2	3	1	0	0	0	0	40	1	1	1	1
4	28	338409	13	0	1	3	0	0	0	0	0	40	0	1	1	1

Después se generan el conjunto de datos para entrenar y evaluar. El método que sigue es exactamente el mismo que el de clasificación.

Ahora definimos los parámetros del multi-layer perceptron regressor. Los valores ingresados fueron iguales a los de clasificación.

```
## definir Multi-layer Perceptron regressor |
mlp_reg = MLPRegressor(
    # definir las capas
    # el numero de neuronas por cada capa
    hidden_layer_sizes=(2),
    # numero maximo de iteraciones
    max_iter = 300,
    # funcion transfer/activation
    activation = 'relu',
    # optimizador de pesos
    solver = 'adam'
)
```

Después se realizó el entrenamiento.

```
# Entrenar el modelo
mlp_reg.fit(X_train_scaled, y_train)

▼
MLPRegressor
MLPRegressor(hidden_layer_sizes=2, max_iter=300)
```

Se pasaron los datos a una variable para ser comparados de igual manera en la que se pasaron en clasificación.

Después se realizó la evaluación.

Para regresión los valores a evaluar fueron los siguientes.

1. R cuadrado (r2): R cuadrado es una medida de qué tan bien se ajusta el modelo a los datos observados. Toma valores entre 0 y 1, donde 1 indica un ajuste perfecto del modelo a los datos y 0 indica que el modelo no explica ninguna variabilidad en los datos. Un valor de r2 de 0.70, por ejemplo, significa que el modelo explica el 70% de la variabilidad en los datos de prueba.

2. Error cuadrático medio (MSE): El MSE es una medida del promedio de los errores al cuadrado entre los valores reales y los valores predichos por el modelo. Cuanto menor sea el valor del MSE, mejor será el ajuste del modelo

a los datos. Un MSE de 0.50, por ejemplo, indica que los errores promedio al cuadrado entre las predicciones y los valores reales son de 0.50 unidades.

3. Varianza explicada (explained variance): La varianza explicada es una medida de cuánta varianza del conjunto de datos de prueba es explicada por el modelo. Toma valores entre 0 y 1, donde 1 indica que el modelo explica toda la varianza y 0 indica que el modelo no explica ninguna varianza. Un valor de 0.80, por ejemplo, significa que el modelo explica el 80% de la varianza en los datos de prueba.

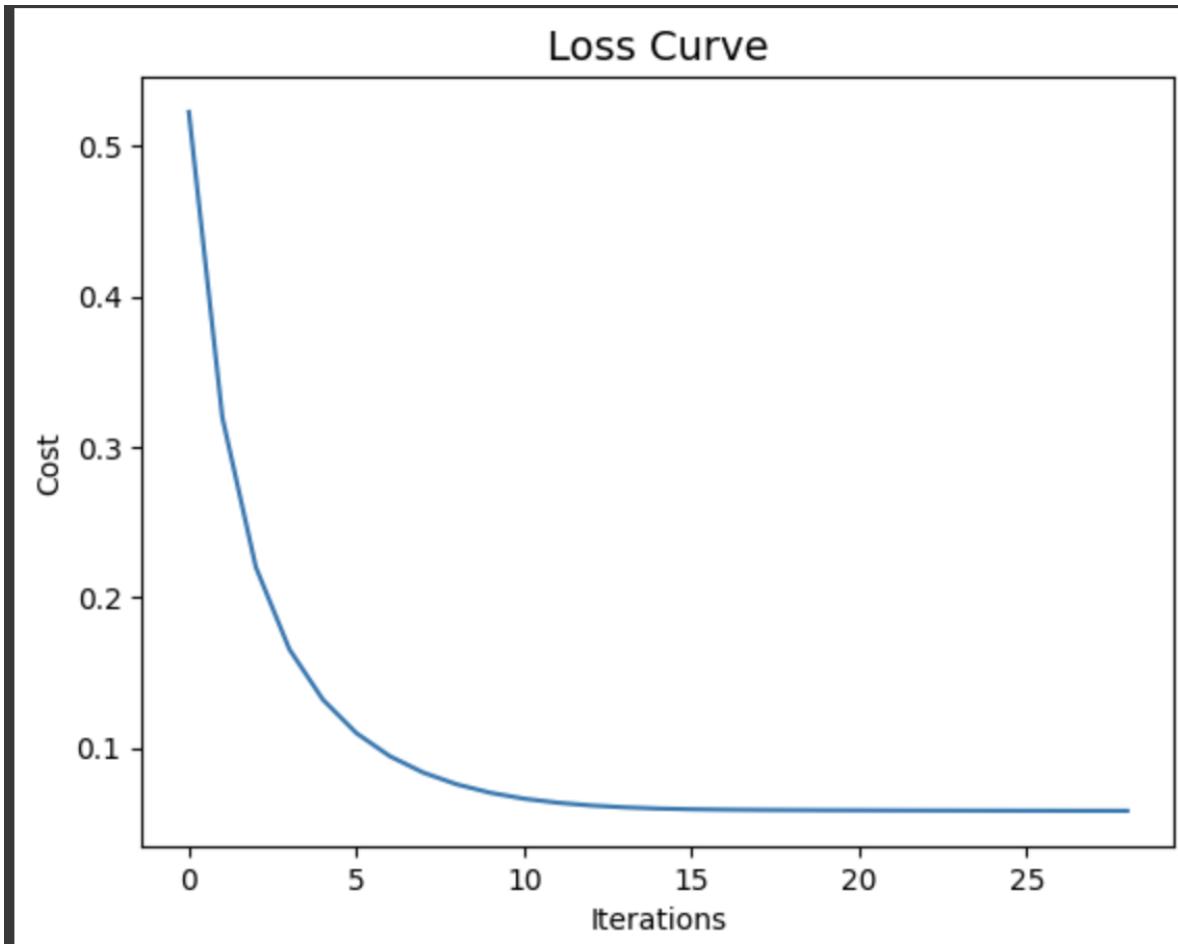
```
# r2
# Mejor valor es 1. Valores pueden ser negativos
print('r2: {:.2f}'.format(r2_score(y_test, y_pred)))

# mean square error
# mejor valor es 0.0
print('mse: {:.2f}'.format(mean_squared_error(y_test, y_pred)))

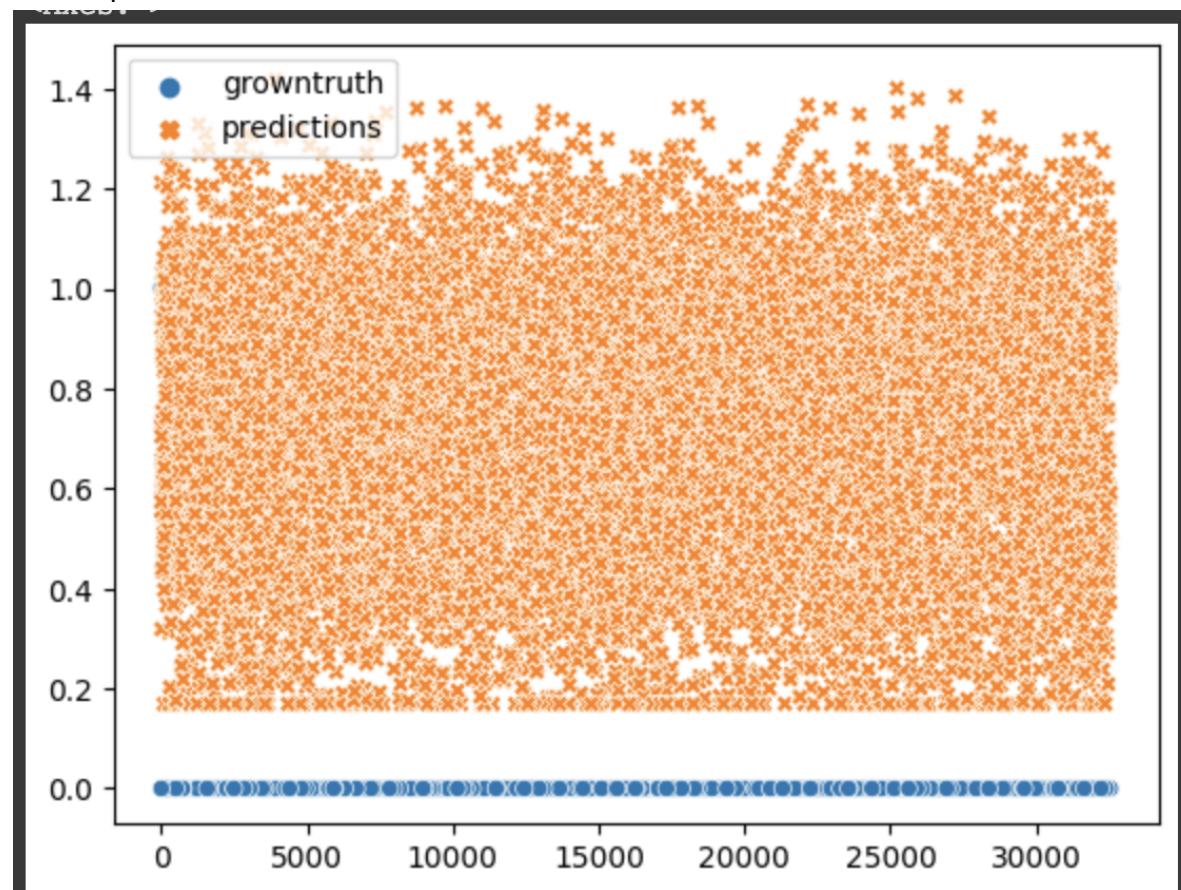
# explained_variance_score
# mejor valor es 1. mientras mas chico, peor
print('explained variance: {:.2f}'.format(explained_variance_score(y_test, y_pred)))

r2: 0.35
mse: 0.12
explained variance: 0.35
```

Loss Curve



Scatter plot



A continuación se generó un scatter plot, el cual nos indica la diferencia entre las predicciones y los valores reales. Como se puede observar no podemos decir que la predicción es precisa.

b. Optimización de hiper-parámetros

Para realizar la regresión también se escogió el mismo acercamiento que fue escogido para clasificación cual es, búsqueda exhaustiva. Los parámetros fueron los siguientes.

```
## Entrenar el modelo con diferentes parametros
#   model
param_grid = {
    'hidden_layer_sizes': [(150,100,50), (120,80), (100)],
    'max_iter': [50, 100, 150],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}
```

Se definió el enfoque de búsqueda cuadrícular para entrenar varios modelos con validación cruzada.

```
grid = GridSearchCV(  
    # estructura del modelo en el que estamos interesados  
    mlp_clf,  
    # hyper parametro que queremos entrenar  
    param_grid_v2,  
    # usar paralelización. -1 = utiliza todos los procesos  
    n_jobs = -1,  
    # número de pliegues utilizados en el enfoque de validación cruzada  
    cv = 5  
)
```

De igual manera que en clasificación, en regresión también tuvimos el problema de no poder ejecutarlo en Google Colab por el tiempo. Por lo tanto se realizó lo mismo para regresión.

c. Red neuronal con mejor desempeño

La red neuronal con el mejor desempeño fue seleccionada de la misma manera que en clasificación.

El mejor modelo contiene los siguientes parámetros.

```
The best hyper parameter values are:  
{'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': 100, 'learning_rate': 'constant', 'max_iter': 50, 'solver': 'adam'}
```

Ya una vez teniendo los valores, estos fueron ingresados para entrenar el modelo con los mejores parámetros y analizar su comportamiento.

```
#Convertir valores para que el programa los pueda reconocer  
  
hidden_layer_sizes = grid.best_params_['hidden_layer_sizes']  
  
if isinstance(hidden_layer_sizes, int):  
    hidden_layer_sizes = (hidden_layer_sizes,)  
  
grid.best_params_['hidden_layer_sizes'] = hidden_layer_sizes  
  
mlp_best = MLPRegressor(**grid.best_params_)  
  
#Entrenar el modelo  
mlp_best.fit(X_train_scaled, y_train)  
  
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Op-  
  warnings.warn(  
    v      MLPRegressor  
    MLPRegressor(alpha=0.05, max_iter=50)
```

d. Métricas de evaluación

El código proporcionado calcula y muestra métricas de evaluación para un modelo de regresión. Calcula y muestra el coeficiente de determinación (r^2), el error cuadrático medio (MSE) y la puntuación de varianza explicada (explained variance score). Estas métricas permiten evaluar el ajuste del

modelo a los datos, el promedio de los errores al cuadrado y la capacidad del modelo para explicar la varianza en los datos.

```
# r2
# Mejor valor es 1. Valores pueden ser negativos
print('r2: {:.2f}'.format(r2_score(y_test, y_pred)))

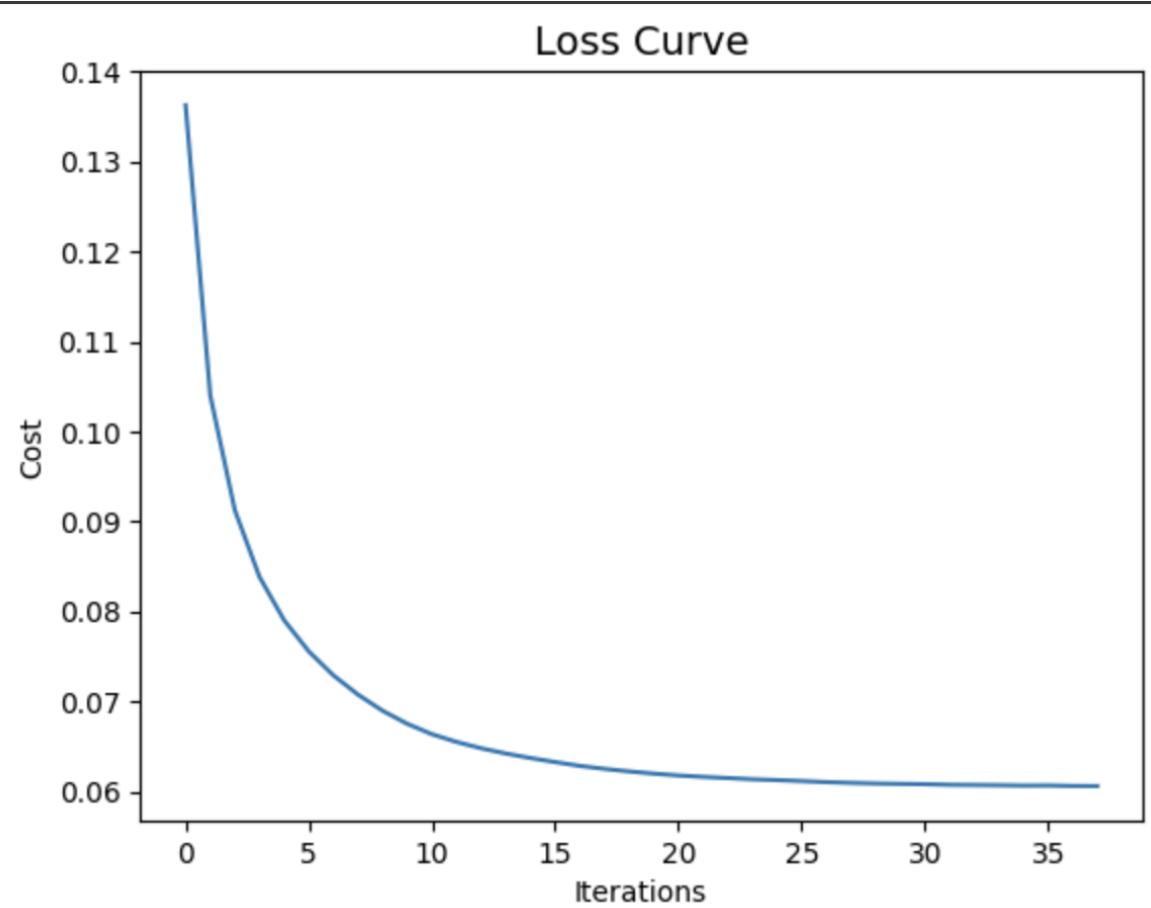
# mean square error
# mejor valor es 0.0
print('mse: {:.2f}'.format(mean_squared_error(y_test, y_pred)))

# explained_variance_score
# mejor valor es 1. mientras mas chico, peor
print('explained variance: {:.2f}'.format(explained_variance_score(y_test, y_pred)))

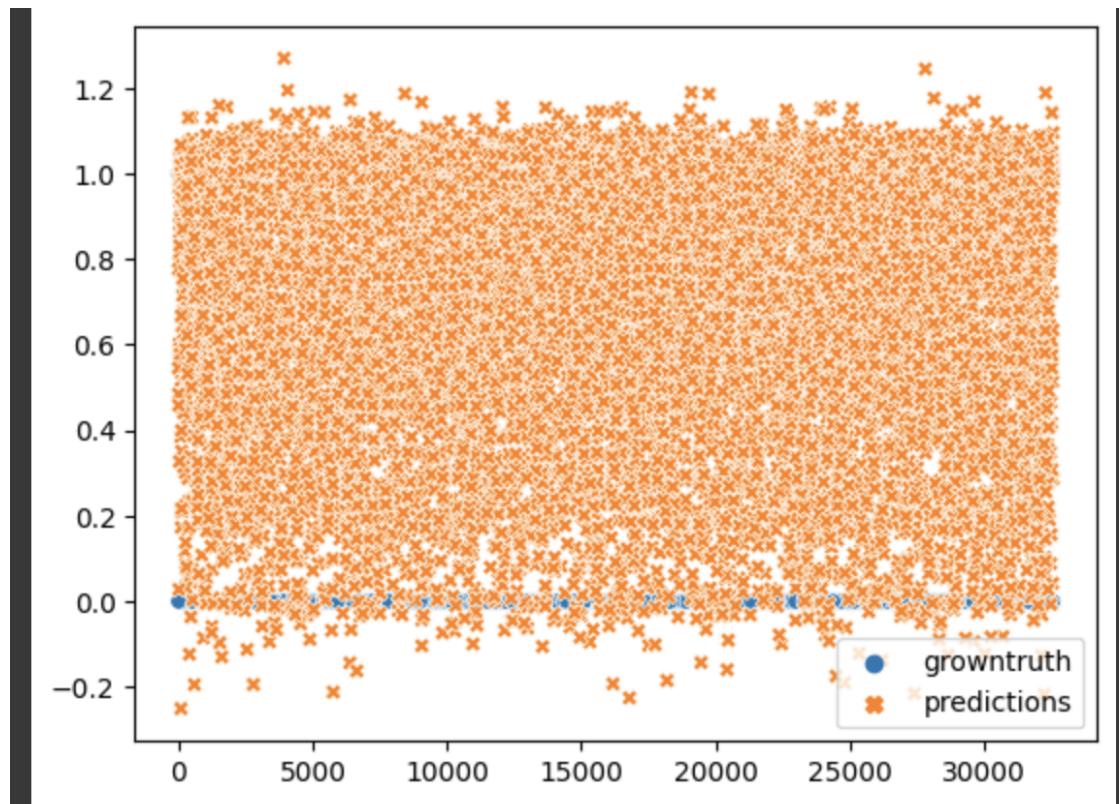
print('-----')
print('Base Model')
print('r2 : ',base_rep_r2)
print('mse : ',base_rep_mse)
print('var : ',base_rep_var)

r2: 0.40
mse: 0.11
explained variance: 0.40
-----
Base Model
r2 : 0.3808676154937761
mse : 0.11657330048865414
var : 0.3808827436593627
```

Loss Curve:



scatter plot: para observar que de acertadas fueron las predicciones.



e. Análisis de los mejores resultados

Desde los resultados obtenidos se puede observar que si existe una diferencia entre el desempeño del mejor modelo y el modelo base. Se puede observar una mejora en el valor de r^2 . En el scatter plot también se puede observar como el modelo comienza a acertarle a ciertos valores reales. Y también pudimos observar una reducción en la varianza. Sin embargo los valores se siguen concentrando en la parte superior del diagrama.

6. Conclusión

Se aprendió sobre la implementación de redes neuronales MLP y la importancia de seleccionar una estructura adecuada. También se fomentó la investigación y la referencia de trabajos previos relevantes para fundamentar las decisiones de implementación. Con base a nuestro proyecto, se observó que con el ajuste de hiperparámetros no tuvo un impacto significativo en el rendimiento del modelo. Además, se observó que el tiempo requerido para entrenar fue tardado en Google Colab, se es necesario buscar formas más eficientes de optimizar hiperparámetros, como el uso de técnicas de optimización más rápidas y eficientes. Esto permitirá reducir los tiempos de entrenamiento y mejorar el rendimiento general del modelo.