Universidad del Valle de Guatemala

Data Science

Julio Avila

Elisa Samayoa

Laboratorio 10

Para este laboratorio, no se especificó que variable debía predecirse, tampoco que base de datos de las dos usar o que modelos debían utilizarse específicamente, por lo que se decidió predecir el nivel de pobreza tanto para hogares como para personas según distintas características encontradas en las bases de datos.

Carga y Limpieza de Datos

Para la carga y limpieza se datos primero se descargaron de canvas, los datos estaban en formato .sav; por lo que tuvo que investigarse como trabajar ese tipo de datos.

Además, se notó que habían demasiados valores faltantes en la base de datos, por lo que fue necesario hacer un proceso de limpieza de los datos para eliminar las columnas que tenían una gran cantidad de datos faltantes y en las que no eran muchos los datos faltantes se colocó la media ya que si se eliminaban todas las filas o columnas con datos faltantes, las bases de datos quedaban demasiado pequeñas.

```python
import pyreadstat

# Cargar datos desde un archivo SPSS
hogar, meta = pyreadstat.read_sav("ENCOVI_Hogar.sav")

# Acceder a los datos y la metadata
print(hogar.head())  # Muestra las primeras filas de datos
#print(hogar)         # Muestra la metadata

# Cargar datos desde un archivo SPSS
personas, meta = pyreadstat.read_sav("ENCOVI_Personas.sav")

# Acceder a los datos y la metadata
print(personas.head())  # Muestra las primeras filas de datos
#print(personas)
```

```
missing_values_hogar = hogar.isnull().sum()
missing_values_personas = personas.isnull().sum()

print("Missing values in 'hogar' data frame:")
print(missing_values_hogar)

print("\nMissing values in 'personas' data frame:")
print(missing_values_personas)
```

```
Missing values in 'hogar' data frame:
REGION        0
DEPTO         0
AREA          0
UPM           0
NUMHOG        0
            ...
P01H15     2641
P01H16     2641
DIA_ENC       0
MES_ENC       1
A_ENC         1
Length: 163, dtype: int64

Missing values in 'personas' data frame:
REGION         0
DEPTO          0
AREA           0
UPM            0
NUMHOG         0
            ...
P11B02B    54247
P11B03A     8999
P11B03B    54612
P11B04A     9001
P11B04B    54549
Length: 465, dtype: int64
```

Aquí se puede observar la carga de datos y donde se encontró la gran cantidad de valores faltantes, el resto del proceso de limpieza y pre procesamiento puede observarse en el notebook ya que fue un proceso bastante extenso.

Después de la limpieza las bases de datos quedaron así:

```
personas_cleaned_final_mean_imputed = personas_cleaned_final.fillna(personas_cle
Final cleaned 'hogar' data frame with mean imputation:
       REGION  DEPTO  AREA    UPM   NUMHOG  FACTOR  FACTOR3  POBREZA  THOGAR  \
0         1.0    1.0   1.0    1.0      1.0   525.0   1575.0      3.0     3.0
1         1.0    1.0   1.0    1.0      2.0   525.0   1575.0      3.0     3.0
2         1.0    1.0   1.0    1.0      3.0   525.0   9450.0      2.0    18.0
3         1.0    1.0   1.0    1.0      4.0   525.0   3150.0      2.0     6.0
4         1.0    1.0   1.0    1.0      5.0   525.0   1575.0      3.0     3.0
...       ...    ...   ...    ...      ...     ...      ...      ...     ...
11531     8.0   17.0   2.0  833.0  11532.0   305.0    915.0      3.0     3.0
11532     8.0   17.0   2.0  833.0  11533.0   305.0   3355.0      1.0    11.0
11533     8.0   17.0   2.0  833.0  11534.0   305.0   1220.0      2.0     4.0
11534     8.0   17.0   2.0  833.0  11535.0   305.0    915.0      3.0     3.0
11535     8.0   17.0   2.0  833.0  11536.0   305.0   1830.0      1.0     6.0

       PPB01  ...    P01H10    P01H11    P01H12    P01H13    P01H14    P01H15  \
0        2.0  ...  1.582125  1.592917  2.006633  1.839236  1.879708  2.047218
1        2.0  ...  1.582125  1.592917  2.006633  1.839236  1.879708  2.047218
2        2.0  ...  1.000000  1.000000  1.000000  1.000000  1.000000  1.000000
3        2.0  ...  2.000000  2.000000  2.000000  2.000000  2.000000  2.000000
4        2.0  ...  1.582125  1.592917  2.006633  1.839236  1.879708  2.047218
...      ...  ...       ...       ...       ...       ...       ...       ...
11531    2.0  ...  2.000000  2.000000  2.000000  2.000000  2.000000  2.000000
11532    2.0  ...  1.000000  1.000000  2.000000  1.000000  1.000000  2.000000
11533    2.0  ...  2.000000  2.000000  2.000000  2.000000  2.000000  2.000000
11534    2.0  ...  2.000000  2.000000  2.000000  2.000000  2.000000  2.000000
11535    2.0  ...  2.000000  2.000000  2.000000  2.000000  2.000000  2.000000

         P01H16  DIA_ENC  MES_ENC   A_ENC
0      2.063631     22.0      8.0  2014.0
1      2.063631     27.0      8.0  2014.0
2      1.000000     22.0      8.0  2014.0
3      2.000000     24.0      8.0  2014.0
4      2.063631     22.0      8.0  2014.0
...         ...      ...      ...     ...
11531  2.000000      5.0     12.0  2014.0
11532  2.000000      5.0     12.0  2014.0
```

	REGION	DEPTO	AREA	UPM	NUMHOG	FACTOR	POBREZA	THOGAR	ID	\
0	1.0	1.0	1.0	1.0	1.0	525.0	3.0	3.0	1.0	
1	1.0	1.0	1.0	1.0	1.0	525.0	3.0	3.0	2.0	
2	1.0	1.0	1.0	1.0	1.0	525.0	3.0	3.0	3.0	
3	1.0	1.0	1.0	1.0	2.0	525.0	3.0	3.0	1.0	
4	1.0	1.0	1.0	1.0	2.0	525.0	3.0	3.0	2.0	
...	...	...	...	...	...	...	...	...	...	
54817	8.0	17.0	2.0	833.0	11536.0	305.0	1.0	6.0	2.0	
54818	8.0	17.0	2.0	833.0	11536.0	305.0	1.0	6.0	3.0	
54819	8.0	17.0	2.0	833.0	11536.0	305.0	1.0	6.0	4.0	
54820	8.0	17.0	2.0	833.0	11536.0	305.0	1.0	6.0	5.0	
54821	8.0	17.0	2.0	833.0	11536.0	305.0	1.0	6.0	6.0	

	PPA02	...	P11A05A	P11A06A	P11A07A	P11A08A	P11A09A	P11A10A	\
0	1.0	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	
1	2.0	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	
2	2.0	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	
3	2.0	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	
4	1.0	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	
...	...	...	...	...	...	...	...	...	
54817	2.0	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	
54818	1.0	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	
54819	2.0	...	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000	
54820	2.0	...	1.945115	1.970561	1.998276	1.992515	1.997359	1.99952	
54821	2.0	...	1.945115	1.970561	1.998276	1.992515	1.997359	1.99952	

	P11B01A	P11B02A	P11B03A	P11B04A
0	2.000000	2.000000	2.000000	2.000000
1	2.000000	2.000000	2.000000	2.000000
2	2.000000	2.000000	2.000000	2.000000
3	2.000000	2.000000	2.000000	2.000000
4	2.000000	2.000000	2.000000	2.000000
...	...	...	...	...
54817	2.000000	2.000000	2.000000	2.000000
54818	2.000000	2.000000	2.000000	2.000000
54819	2.000000	2.000000	2.000000	2.000000
54820	1.965345	1.987452	1.995417	1.994042
54821	1.965345	1.987452	1.995417	1.994042

[54822 rows x 100 columns]

Tras este proceso se obtuvieron las variables con una correlación con la variable POBREZA mayor o igual al valor absoluto de 0.1 para ser las variables con las que se trabajaría en el análisis estadístico y posteriormente elegir cuales se incluirían en los modelos.

```
# Calcular correlación en la base de datos 'hogar'
correlation_hogar = hogar_cleaned_final_mean_imputed.corr()['POBREZA']

# Filtrar variables con correlación mayor o igual a 0.5 en 'hogar'
high_corr_variables_hogar = correlation_hogar[abs(correlation_hogar) >= 0.1].index.tolist()

# Imprimir lista de variables con correlación mayor o igual a 0.1 en 'hogar'
print("Variables en 'hogar' con correlación mayor o igual a 0.1 con 'POBREZA':")
print(high_corr_variables_hogar)

# Calcular correlación en la base de datos 'personas'
correlation_personas = personas_cleaned_final_mean_imputed.corr()['POBREZA']

# Filtrar variables con correlación mayor o igual a 0.5 en 'personas'
high_corr_variables_personas = correlation_personas[abs(correlation_personas) >= 0.1].index.tolist()

# Imprimir lista de variables con correlación mayor o igual a 0.1 en 'personas'
print("\nVariables en 'personas' con correlación mayor o igual a 0.1 con 'POBREZA':")
print(high_corr_variables_personas)
```

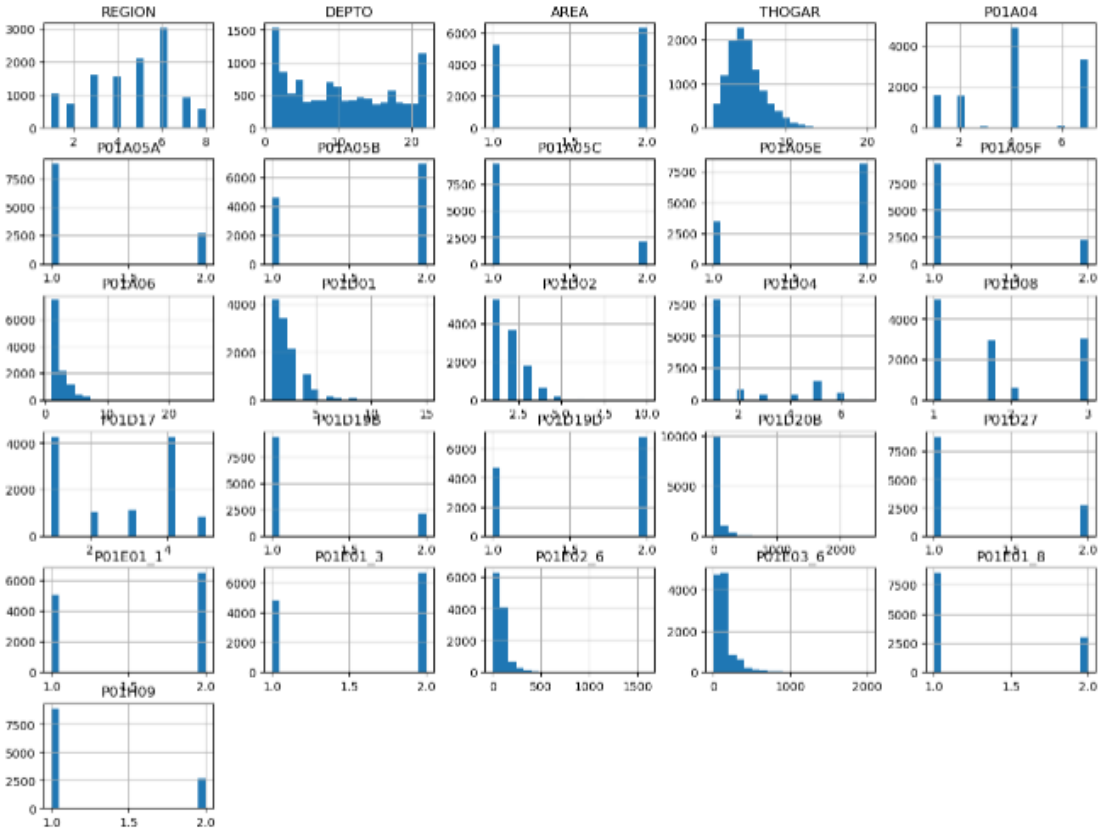
```
C:\Users\USUARIO\AppData\Local\Temp\ipykernel_1344\2691544929.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to
False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_hogar = hogar_cleaned_final_mean_imputed.corr()['POBREZA']
Variables en 'hogar' con correlación mayor o igual a 0.1 con 'POBREZA':
['REGION', 'DEPTO', 'AREA', 'UPM', 'NUMHOG', 'FACTOR3', 'POBREZA', 'THOGAR', 'PPB04', 'PPD04A', 'PPD06', 'PPD07', 'PPD08', 'P01A02', 'P01A04', 'P01A05A', 'P01A05B', 'P01A05C', 'P01A05D', 'P0
1A05E', 'P01A05F', 'P01A06', 'P01B02', 'P01D01', 'P01D02', 'P01D04', 'P01D08', 'P01D09', 'P01D17', 'P01D19A', 'P01D19B', 'P01D19C', 'P01D19D', 'P01D20B', 'P01D23A', 'P01D27', 'P01E01_1', 'P0
1E01_2', 'P01E01_3', 'P01E01_4', 'P01E01_5', 'P01E01_6', 'P01E02_6', 'P01E03_6', 'P01E01_8', 'P01H09']
C:\Users\USUARIO\AppData\Local\Temp\ipykernel_1344\2691544929.py:12: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to
False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_personas = personas_cleaned_final_mean_imputed.corr()['POBREZA']

Variables en 'personas' con correlación mayor o igual a 0.1 con 'POBREZA':
['REGION', 'DEPTO', 'AREA', 'UPM', 'NUMHOG', 'POBREZA', 'THOGAR', 'ID', 'PPA03', 'PPA04C', 'PPA06', 'PPA08', 'P04A01A', 'P04A04A', 'P04A07A', 'P04A08A', 'P04A11A', 'P06B01', 'P06B03', 'P06B2
5A', 'P06B26', 'P08A01A', 'P08A02', 'P09C08A', 'P09C09A', 'P09E01A', 'P09F02A', 'P09F08A', 'P09F09A', 'P10A01']
```

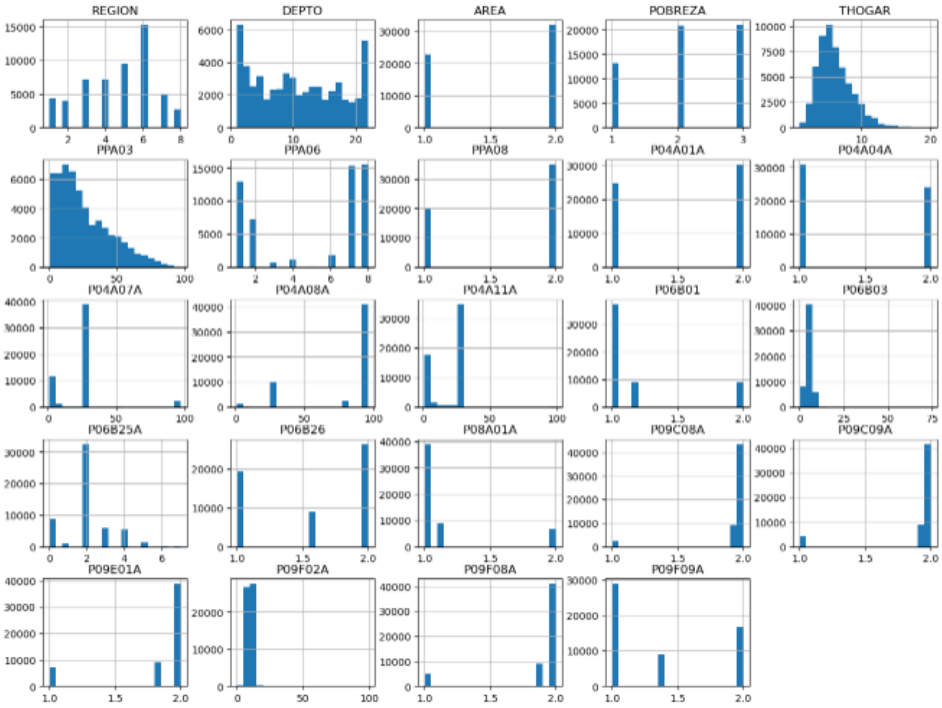
Análisis Univariable

Se buscó la distribución de las variables que cumplían con ese parámetro de correlación para ir viendo el comportamiento de las variables y así poder ir descartando cuales no iban a servir en los modelos.

Hogares:



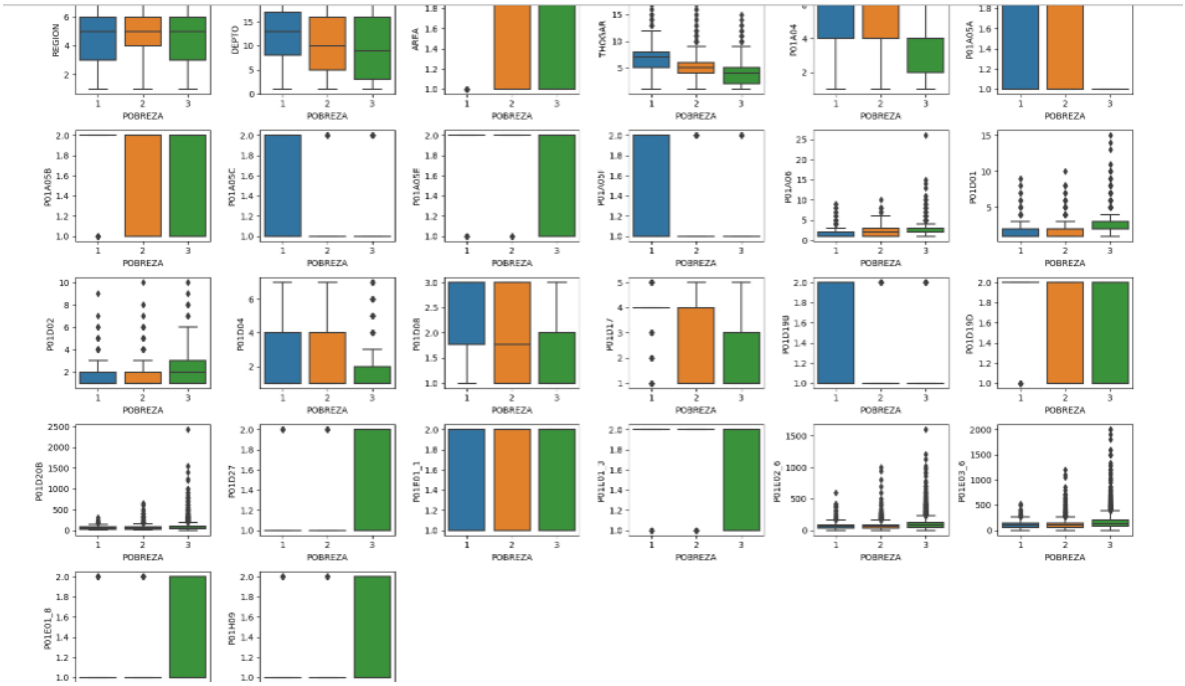
Personas:



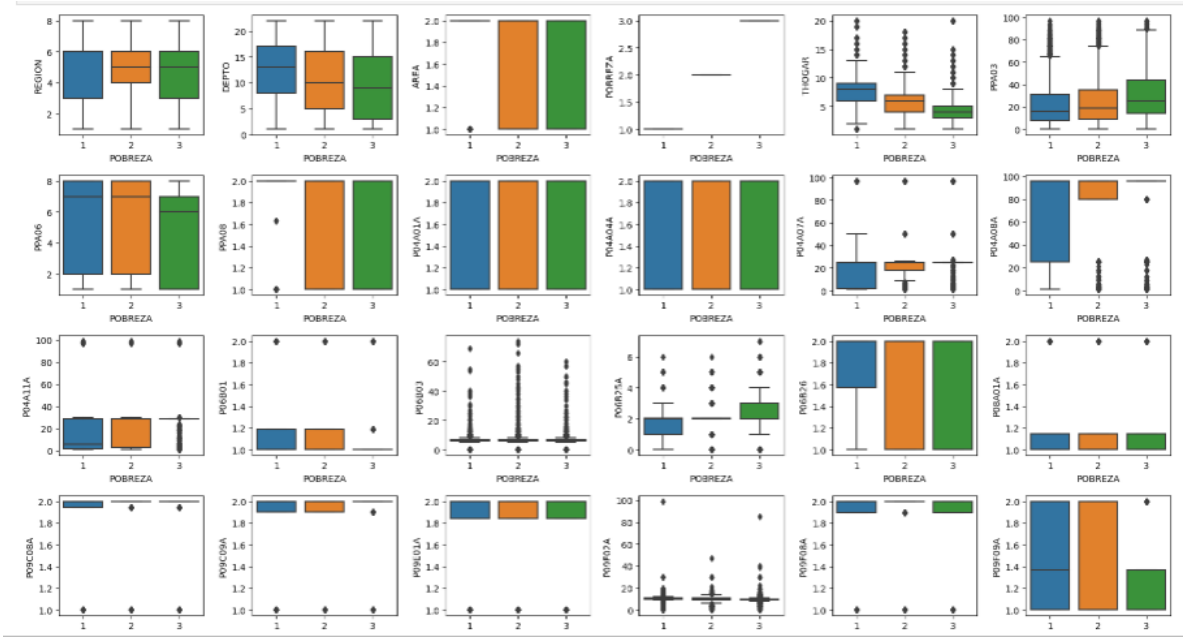
Análisis Bivariable

Posteriormente se analizó mediante boxplots el comportamiento de las variables respecto a la variable a predecir (POBREZA) y así seleccionar finalmente cuales serían las que se usarían en el modelo

Hogares:



Personas:



Preparación de Datos para Modelado:

Así se decidió que las variables a utilizar en los modelos serían:

Después de hacer el análisis estadístico de las variables con una correlación considerable, elegimos las variables a usar en los modelos.

En el modelo de hogar, serían:

- Region
- Depto
- Area
- THOGAR
- P01A04
- P01A05A
- P01A05B
- P01A06
- P01D01
- P01D02
- P01D04
- P01D08
- P01D17
- P01D19D
- P01D20B

- P01E01_1
- P01E02_6
- P01E03_6

y en el modelo de personas serían:

- REGION
- DEPTO
- AREA
- THOGAR
- PPA03
- PPA06
- PPA08
- P04A01A
- P04A04A
- P04A07A
- P04A08A
- P04A11A
- P06B01
- P06B03
- P06B25A
- P06B26
- P08A01A
- P09C08A
- P09C09A
- P09E01A
- P09F02A
- P09F08A
- P09F09A

Además se hizo la división de los datos en train y test

Modelos de hogares:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from math import sqrt
```

```
# Seleccionar las variables predictoras
selected_features = ['REGION', 'DEPTO', 'THOGAR', 'P01A04', 'P01A05A', 'P01A05B', 'P01A06',
                     'P01D01', 'P01D02', 'P01D04', 'P01D08', 'P01D17', 'P01D19D', 'P01D20B',
                     'P01E02_6', 'P01E03_6']

# Seleccionar la variable objetivo
target_variable = 'POBREZA'

# Filtrar el DataFrame con las columnas seleccionadas
df_hogar_selected = hogar_cleaned_final_mean_imputed[selected_features + [target_variable]].dropna()

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(df_hogar_selected[selected_features],
                                                    df_hogar_selected[target_variable],
                                                    test_size=0.2, random_state=42)
```

Modelos de personas:

69]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from math import sqrt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

70]:
```
# Seleccionar las variables predictoras
selected_features_personas = ['REGION', 'DEPTO', 'AREA', 'THOGAR', 'PPA03', 'PPA06', 'PPA08', 'P04A01A', 'P04A04A',
                              'P04A07A', 'P04A08A', 'P04A11A', 'P06B01', 'P06B03', 'P06B25A', 'P06B26', 'P08A01A',
                              'P09C08A', 'P09C09A', 'P09E01A', 'P09F02A', 'P09F08A', 'P09F09A']

# Seleccionar la variable objetivo
target_variable_personas = 'POBREZA'

# Filtrar el DataFrame con las columnas seleccionadas y eliminar filas con valores nulos
df_personas_selected = personas_cleaned_final_mean_imputed[selected_features_personas + [target_variable_personas]].dropna()

# Dividir el conjunto de datos en entrenamiento y prueba
X_train_personas, X_test_personas, y_train_personas, y_test_personas = train_test_split(
    df_personas_selected[selected_features_personas], df_personas_selected[target_variable_personas],
    test_size=0.2, random_state=42
)

# Normalizar los datos
scaler_personas = StandardScaler()
X_train_normalized_personas = scaler_personas.fit_transform(X_train_personas)
X_test_normalized_personas = scaler_personas.transform(X_test_personas)
```

Construcción del Modelo

Se realizaron para ambas bases de datos modelos de regresión, random forest y una red neuronal

Modelos hogares:

```python
# Modelo de Regresión Lineal
regression_model = LinearRegression()
regression_model.fit(X_train, y_train)
regression_predictions = regression_model.predict(X_test)
regression_rmse = sqrt(mean_squared_error(y_test, regression_predictions))
print(f"Regresión Lineal RMSE: {regression_rmse}")
```

```
Regresión Lineal RMSE: 0.514009442541057
```

```python
# Modelo Random Forest
random_forest_model = RandomForestRegressor()
random_forest_model.fit(X_train, y_train)
rf_predictions = random_forest_model.predict(X_test)
rf_rmse = sqrt(mean_squared_error(y_test, rf_predictions))
print(f"Random Forest RMSE: {rf_rmse}")
```

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from tensorflow.keras.optimizers import Adam

# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(hogar_cleaned_final_mean_imputed[['REGION', 'DEPTO', 'AREA', 'THOGAR',
                        'P01A04', 'P01A05A', 'P01A05B', 'P01A05C',
                       'P01A05E', 'P01A05F', 'P01A06', 'P01D01', 'P01D02', 'P01D04', 'P01D08',
                       'P01D17', 'P01D19D', 'P01D20B', 'P01E01_1', 'P01E02_6', 'P01E03_6']], hogar_cleaned_final_mean_imputed['POBREZA'], test_size=0.2, random_state=42)

# Normalizar los datos
X_train_normalized = (X_train - X_train.mean()) / X_train.std()
X_test_normalized = (X_test - X_train.mean()) / X_train.std()

# Construir el modelo de red neuronal
model = Sequential()
model.add(Dense(64, input_dim=X_train_normalized.shape[1], activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='linear'))

model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')

# Entrenar el modelo
model.fit(X_train_normalized, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Evaluar el modelo en el conjunto de prueba
nn_predictions = model.predict(X_test_normalized)
nn_rmse = sqrt(mean_squared_error(y_test, nn_predictions))
print(f"Red Neuronal RMSE en conjunto de prueba: {nn_rmse}")
```

Modelos personas

```python
# Modelo de Regresión Lineal
regression_model_personas = LinearRegression()
regression_model_personas.fit(X_train_normalized_personas, y_train_personas)
regression_predictions_personas = regression_model_personas.predict(X_test_normalized_personas)
regression_rmse_personas = sqrt(mean_squared_error(y_test_personas, regression_predictions_personas))
print(f"Regresión Lineal RMSE para personas: {regression_rmse_personas}")
```

```
Regresión Lineal RMSE para personas: 0.5964444889804388
```

```python
# Modelo Random Forest
random_forest_model_personas = RandomForestRegressor()
random_forest_model_personas.fit(X_train_normalized_personas, y_train_personas)
rf_predictions_personas = random_forest_model_personas.predict(X_test_normalized_personas)
rf_rmse_personas = sqrt(mean_squared_error(y_test_personas, rf_predictions_personas))
print(f"Random Forest RMSE para personas: {rf_rmse_personas}")
```

```
Random Forest RMSE para personas: 0.5448930320541682
```

```
# Modelo de Red Neuronal
model_personas = Sequential()
model_personas.add(Dense(64, input_dim=X_train_normalized_personas.shape[1], activation='relu'))
model_personas.add(Dropout(0.5))
model_personas.add(Dense(32, activation='relu'))
model_personas.add(Dropout(0.5))
model_personas.add(Dense(1, activation='linear'))

model_personas.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')

# Entrenar el modelo
model_personas.fit(X_train_normalized_personas, y_train_personas, epochs=50, batch_size=32, validation_split=0.2)

# Evaluar el modelo en el conjunto de prueba
nn_predictions_personas = model_personas.predict(X_test_normalized_personas)
nn_rmse_personas = sqrt(mean_squared_error(y_test_personas, nn_predictions_personas))
print(f"Red Neuronal RMSE para personas: {nn_rmse_personas}")
```

Evaluación del Modelo:

Se hizo la comparación entre los 3 modelos de cada una de las bases de datos, llegando a concluirse que en ambos casos el mejor modelo para predecir POBREZA es mediante random forests ya que obtuvieron los valores de rmse mas bajos.

Hogares:

```
# Mostrar resultados
print(f"Regresión Lineal RMSE: {regression_rmse}")
print(f"Random Forest RMSE: {rf_rmse}")
print(f"Red Neuronal RMSE: {nn_rmse}")
```

```
Regresión Lineal RMSE: 0.514009442541057
Random Forest RMSE: 0.4950994511744218
Red Neuronal RMSE: 0.5050387118355069
```

Personas:

```
# Imprimir resultados
print(f"Regresión Lineal RMSE para personas: {regression_rmse_personas}")
print(f"Random Forest RMSE para personas: {rf_rmse_personas}")
print(f"Red Neuronal RMSE para personas: {nn_rmse_personas}")
```

```
Regresión Lineal RMSE para personas: 0.5964444889804388
Random Forest RMSE para personas: 0.5448930320541682
Red Neuronal RMSE para personas: 0.5775876367919405
```

Debe recalcarse que se probó con diferentes parámetros e hiperparámetros hasta llegar a estos que fueron los mejores valores de rmse, también se intentaron otros tipos de redes y regresiones, sin embargo, no se obtuvieron resultados tan buenos como los seleccionados. A continuación, se pueden ver resultados de algunos modelos que no llegaron a considerarse como los más eficientes para resolver este problema:

```python
# Crear y entrenar el modelo de árbol de decisión
model_tree = DecisionTreeClassifier(random_state=42)
model_tree.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
predictions_tree = model_tree.predict(X_test)

# Evaluar el modelo
accuracy_tree = accuracy_score(y_test, predictions_tree)
print(f'Test accuracy (Decision Tree): {accuracy_tree:.4f}\n')

# Mostrar el informe de clasificación
print('Classification Report (Decision Tree):\n')
print(classification_report(y_test, predictions_tree))
```

```
Test accuracy (Decision Tree): 0.6438

Classification Report (Decision Tree):

              precision    recall  f1-score   support

           0       0.51      0.55      0.53       364
           1       0.56      0.53      0.54       875
           2       0.76      0.77      0.76      1069

    accuracy                           0.64      2308
   macro avg       0.61      0.62      0.61      2308
weighted avg       0.64      0.64      0.64      2308
```

```
Epoch 20/20
462/462 [==============================] - 2s 3ms/step - loss: 0.6835 - accuracy: 0.6855 - val_loss: 0.7095 - val_accuracy: 0.6777
73/73 [==============================] - 0s 2ms/step - loss: 0.6836 - accuracy: 0.6902

Test accuracy: 0.6902079582214355
```

Reflexión:

Este laboratorio fue bastante retador debido a que se tuvo mucha libertad para seleccionar modelos, variables a predecir y análisis a realizar. Permite poner en práctica un rango más amplio de conocimientos adquiridos a lo largo del curso.

Otra de las principales dificultades para realizar este laboratorio fue la gran cantidad de procesamiento que los datos requerían ya que mostraban una cantidad grandísima de datos faltantes.

Este es el laboratorio donde se ha tenido la mayor libertad en cuanto a las decisiones a tomar y, por lo mismo, da la oportunidad de experimentar con muchas cosas y encontrar así los modelos más eficientes posibles. Esto permite también conocer un poco más como funcionan, posiblemente escenarios reales en un ambiente laboral ya que es muy común que se soliciten trabajos o resultados sin una gran explicación o guía de como hacerlo y como científicos de datos es nuestro trabajo encontrar la mejor forma de resolver dichos problemas.