Análisis de ENAHO-Módulo de Salud

PEA DATA SCIENCE

Edizon Colquichagua Valer

•¿Qué es ENAHO?

la ENAHO es una herramienta fundamental para monitorear y analizar las condiciones de vida de la población peruana, proporcionando datos clave para la toma de decisiones en políticas públicas y programas sociales.

•¿Para qué Sirve?

La Encuesta Nacional de Hogares (ENAHO) tiene como objetivo principal generar indicadores que permitan conocer la evolución de la pobreza, el bienestar y las condiciones de vida de los hogares en el Perú. Se realiza de manera continua desde mayo de 2003 y abarca tanto el área urbana como rural en los 24 departamentos del país y en la Provincia Constitucional del Callao.

Esta encuesta se lleva a cabo para efectuar diagnósticos sobre las condiciones de vida y pobreza de la población, medir el impacto de los programas sociales en la mejora de las condiciones de vida, servir como fuente de información para instituciones públicas y privadas, así como para investigadores, y permitir la comparabilidad con investigaciones similares en relación con las variables investigadas.

Metodología de muestreo y cantidad de muestra que utiliza.

La Encuesta Nacional de Hogares (ENAHO) utiliza una metodología de muestreo que permite obtener estimaciones representativas de la población. La muestra se distribuye en conglomerados de viviendas en los 24 departamentos del país y en la Provincia Constitucional del Callao. En total, la muestra de la ENAHO 2023 consta de 36,726 viviendas distribuidas en 5359 conglomerados.

La muestra se diseña para obtener estimaciones a nivel nacional, urbano y rural, así como para cada uno de los departamentos como dominios de estudio. Además, se consideran diferentes niveles de inferencia, como la muestra integrada (panel y no panel) y la muestra panel, permitiendo producir resultados para diversos arreglos de unidades y niveles de desagregación.

En resumen, la ENAHO utiliza una muestra representativa y diversificada que garantiza la precisión en las estimaciones de las características socio-demográficas de la población en diferentes niveles geográficos y contextos.

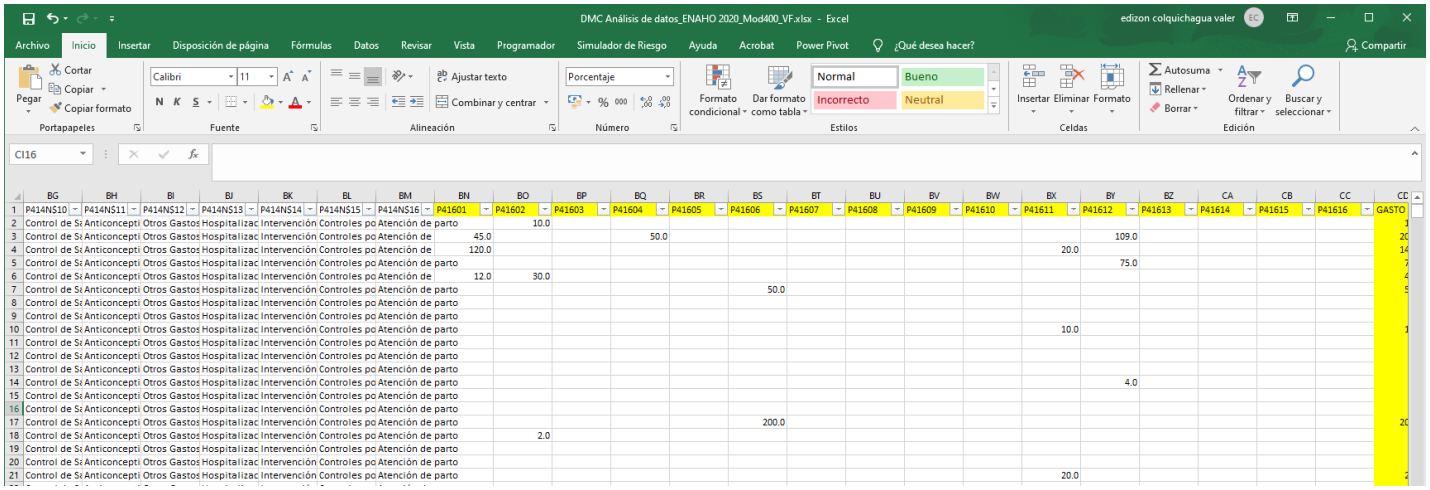
Las variables que seleccionaremos son:

Variables independientes:

Edad (P208A)
Nivel de educación (P301A)
Sexo (P207)
Enfermedad crónica (P401)

Variable dependiente:

Gasto total



Informe Resumido: Análisis y Modelado de Datos

1. Limpieza de Datos:

Cargamos el conjunto de datos y seleccionamos las variables de interés ('P208A', 'P301A', 'P207', 'P401', 'GASTO TOTAL').
Eliminamos las filas con valores nulos en la variable 'GASTO TOTAL'.
Aplicamos transformaciones necesarias, como codificación one-hot para variables categóricas ('P301A', 'P207', 'P401').

```
# Estadísticas adicionales
print("\nEstadísticas adicionales de GASTO TOTAL:")
print("Media:", df_clean['GASTO TOTAL'].mean())
print("Mediana:", df_clean['GASTO TOTAL'].median())
print("Mínimo:", df_clean['GASTO TOTAL'].min())
print("Máximo:", df_clean['GASTO TOTAL'].max())
print("Cuartiles:")
print(df_clean['GASTO TOTAL'].quantile([0.25, 0.5, 0.75]))
```

```
Estadísticas adicionales de GASTO TOTAL:
Media: 309.00106607737
Mediana: 0.0
Mínimo: 0.0
Máximo: 199999.8
Cuartiles:
0.25     0.0
0.50     0.0
0.75    15.0
Name: GASTO TOTAL, dtype: float64
```

```
# Contar valores cero en 'GASTO TOTAL'
num_zeros = (df_clean['GASTO TOTAL'] == 0).sum()
print("Número de valores cero en 'GASTO TOTAL':", num_zeros)
```

```
Número de valores cero en 'GASTO TOTAL': 10490
```

```
# Eliminar filas con valores cero en 'GASTO TOTAL'
df_filtered = df_clean[df_clean['GASTO TOTAL'] != 0]

# Mostrar las primeras filas del DataFrame filtrado
print(df_filtered.head())
```

```
   P208A                              P301A    P207 P401  GASTO TOTAL
0   50.0  Superior Universitaria Incompleta  Hombre   Si         10.0
1   44.0  Superior Universitaria Incompleta   Mujer   No        204.0
2   24.0  Superior Universitaria Incompleta   Mujer   Si        140.0
3   21.0  Superior Universitaria Incompleta   Mujer   Si         75.0
4   44.0               Secundaria Completa  Hombre   No         42.0
```

```
# Boxplot de la columna 'GASTO TOTAL' sin valores cero
plt.figure(figsize=(8, 6))
sns.boxplot(data=df_filtered, y='GASTO TOTAL')
plt.title('Boxplot de GASTO TOTAL sin valores cero')
plt.ylabel('GASTO TOTAL')
plt.show()

# Estadísticas adicionales del DataFrame filtrado
print("\nEstadísticas adicionales de GASTO TOTAL sin valores cero:")
print("Media:", df_filtered['GASTO TOTAL'].mean())
print("Mediana:", df_filtered['GASTO TOTAL'].median())
print("Mínimo:", df_filtered['GASTO TOTAL'].min())
print("Máximo:", df_filtered['GASTO TOTAL'].max())
print("Cuartiles:")
print(df_filtered['GASTO TOTAL'].quantile([0.25, 0.5, 0.75]))
```

```
# Identificar las variables categóricas
variables_categoricas = ['P301A', 'P207', 'P401']
```

```
# Copiar el DataFrame filtrado dentro del rango intercuartílico
df_transformado = df_filtered_within_iqr.copy()
# Transformar variables categóricas dicotómicas
# Supongamos que 'P207' y 'P401' son variables categóricas dicotómicas
dicotomicas = {'P207': {'Hombre': 1, 'Mujer': 0}, 'P401': {'Si': 1, 'No': 0}}
df_transformado.replace(dicotomicas, inplace=True)
# Transformar variables categóricas con más de dos categorías utilizando one-hot encoding
# Supongamos que 'P301A' es una variable categórica con más de dos categorías
df_transformado = pd.get_dummies(df_transformado, columns=['P301A'])
```

```
# Mostrar las primeras filas del DataFrame después de aplicar las transformaciones
print("\nMuestras del DataFrame después de las transformaciones:")
print(df_transformado.head())
```

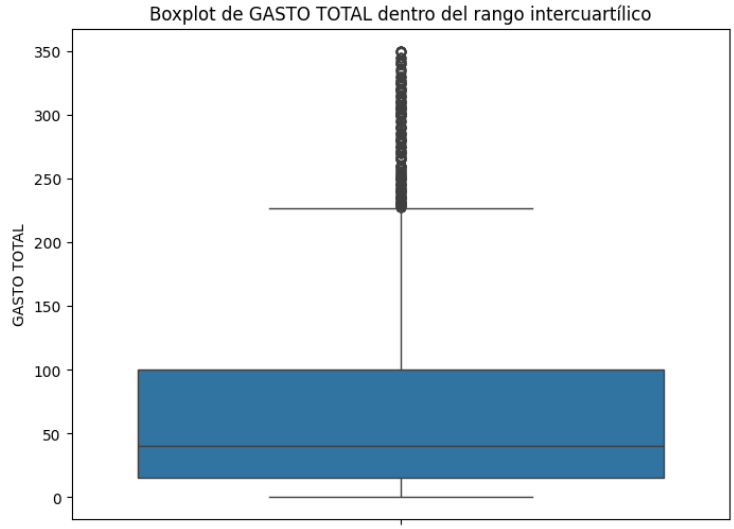
2. Exploración de Datos:

Realizamos análisis descriptivos básicos para comprender la distribución de las variables y detectar valores atípicos.

```python
# Calcular el rango intercuartílico (IQR)
Q1 = df_filtered['GASTO TOTAL'].quantile(0.25)
Q3 = df_filtered['GASTO TOTAL'].quantile(0.75)
IQR = Q3 - Q1

# Definir los límites inferior y superior
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filtrar los valores dentro del rango intercuartílico
df_filtered_within_iqr = df_filtered[(df_filtered['GASTO TOTAL'] >= lower_bound) & (df_filtered['GASTO TOTAL'] <= upper_bound)]

# Mostrar las primeras filas del DataFrame con valores dentro del rango intercuartílico
print(df_filtered_within_iqr.head())
```

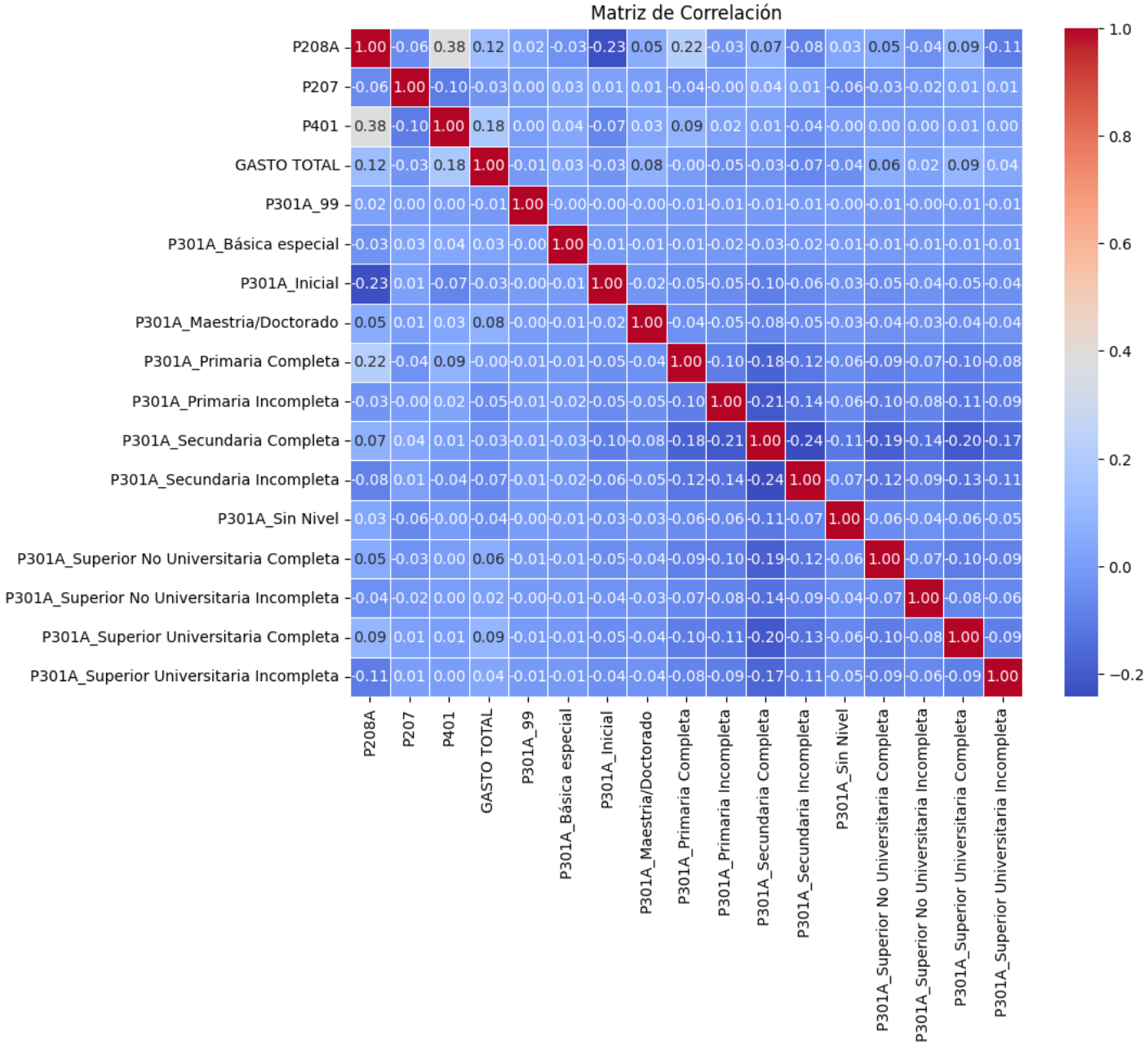
```
   P208A                          P301A    P207 P401  GASTO TOTAL
0   50.0  Superior Universitaria Incompleta  Hombre   Si         10.0
1   44.0  Superior Universitaria Incompleta   Mujer   No        204.0
2   24.0  Superior Universitaria Incompleta   Mujer   Si        140.0
3   21.0  Superior Universitaria Incompleta   Mujer   Si         75.0
4   44.0               Secundaria Completa  Hombre   No         42.0
```

```python
# Boxplot de la columna 'GASTO TOTAL' dentro del rango intercuartílico
plt.figure(figsize=(8, 6))
sns.boxplot(data=df_filtered_within_iqr, y='GASTO TOTAL')
plt.title('Boxplot de GASTO TOTAL dentro del rango intercuartílico')
plt.ylabel('GASTO TOTAL')
plt.show()

# Estadísticas adicionales del DataFrame filtrado dentro del rango intercuartílico
print("\nEstadísticas adicionales de GASTO TOTAL dentro del rango intercuartílico:")
print("Media:", df_filtered_within_iqr['GASTO TOTAL'].mean())
print("Mediana:", df_filtered_within_iqr['GASTO TOTAL'].median())
print("Mínimo:", df_filtered_within_iqr['GASTO TOTAL'].min())
print("Máximo:", df_filtered_within_iqr['GASTO TOTAL'].max())
print("Cuartiles:")
print(df_filtered_within_iqr['GASTO TOTAL'].quantile([0.25, 0.5, 0.75]))

# Mostrar cantidad de filas en el DataFrame filtrado dentro del rango intercuartílico
num_rows_within_iqr = len(df_filtered_within_iqr)
print("\nCantidad de filas dentro del rango intercuartílico:", num_rows_within_iqr)
```




```
Estadísticas adicionales de GASTO TOTAL dentro del rango intercuartílico:
Media: 73.28553555636643
Mediana: 40.0
Mínimo: 0.20000000298023224
Máximo: 350.0
Cuartiles:
0.25     15.000000
0.50     40.000000
0.75    100.199999
Name: GASTO TOTAL, dtype: float64

Cantidad de filas dentro del rango intercuartílico: 4556
```

```python
# Crear el gráfico de la matriz de correlación
plt.figure(figsize=(10, 8))
sns.heatmap(matriz_correlacion, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Matriz de Correlación')
plt.show()
```



3. Preparación de Datos:

Dividimos el conjunto de datos en variables independientes (X) y variable dependiente (y).
Dividimos los datos en conjuntos de entrenamiento y prueba.

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Definir las variables independientes (X) y la variable dependiente (y)
X = df_transformado[['P208A', 'P207', 'P401', 'P301A_Básica especial', 'P301A_Inicial', 'P301A_Maestria/Doctorado',
                     'P301A_Primaria Completa', 'P301A_Primaria Incompleta', 'P301A_Secundaria Completa',
                     'P301A_Secundaria Incompleta', 'P301A_Sin Nivel', 'P301A_Superior No Universitaria Completa',
                     'P301A_Superior No Universitaria Incompleta', 'P301A_Superior Universitaria Completa',
                     'P301A_Superior Universitaria Incompleta']]
y = df_transformado['GASTO TOTAL']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Instanciar el modelo de regresión lineal
modelo_regresion_lineal = LinearRegression()

# Entrenar el modelo con los datos de entrenamiento
modelo_regresion_lineal.fit(X_train, y_train)

# Realizar predicciones utilizando los datos de prueba
predicciones = modelo_regresion_lineal.predict(X_test)

# Evaluar el desempeño del modelo
mse = mean_squared_error(y_test, predicciones)
r2 = r2_score(y_test, predicciones)
mae = mean_absolute_error(y_test, predicciones)

print("Error Cuadrático Medio (MSE):", mse)
print("Coeficiente de Determinación (R^2):", r2)
print("Error Absoluto Medio (MAE):", mae)
```

4. Modelado:

Probamos varios modelos de regresión, incluyendo regresión lineal, regresión polinómica, regresión de árboles de decisión, bosques aleatorios y redes neuronales.
Evaluamos el desempeño de cada modelo utilizando métricas como el error cuadrático medio (MSE), el coeficiente de determinación (R^2) y el error absoluto medio (MAE).

```python
from sklearn.tree import DecisionTreeRegressor

# Instanciar el modelo de regresión de árboles de decisión
modelo_arboles_decision = DecisionTreeRegressor(random_state=42)

# Entrenar el modelo con los datos de entrenamiento
modelo_arboles_decision.fit(X_train, y_train)

# Realizar predicciones utilizando los datos de prueba
predicciones_arboles_decision = modelo_arboles_decision.predict(X_test)

# Evaluar el desempeño del modelo
mse_arboles_decision = mean_squared_error(y_test, predicciones_arboles_decision)
r2_arboles_decision = r2_score(y_test, predicciones_arboles_decision)
mae_arboles_decision = mean_absolute_error(y_test, predicciones_arboles_decision)

print("Error Cuadrático Medio (MSE) - Árboles de Decisión:", mse_arboles_decision)
print("Coeficiente de Determinación (R^2) - Árboles de Decisión:", r2_arboles_decision)
print("Error Absoluto Medio (MAE) - Árboles de Decisión:", mae_arboles_decision)
```

```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

# Definir el grado del polinomio
grado_polinomio = 2  # Puedes ajustar este valor según lo consideres necesario

# Instanciar el modelo de regresión polinómica
modelo_regresion_polinomial = make_pipeline(PolynomialFeatures(grado_polinomio), LinearRegression())

# Entrenar el modelo con los datos de entrenamiento
modelo_regresion_polinomial.fit(X_train, y_train)

# Realizar predicciones utilizando los datos de prueba
predicciones_regresion_polinomial = modelo_regresion_polinomial.predict(X_test)

# Evaluar el desempeño del modelo
mse_regresion_polinomial = mean_squared_error(y_test, predicciones_regresion_polinomial)
r2_regresion_polinomial = r2_score(y_test, predicciones_regresion_polinomial)
mae_regresion_polinomial = mean_absolute_error(y_test, predicciones_regresion_polinomial)

print("Error Cuadrático Medio (MSE) - Regresión Polinomial:", mse_regresion_polinomial)
print("Coeficiente de Determinación (R^2) - Regresión Polinomial:", r2_regresion_polinomial)
print("Error Absoluto Medio (MAE) - Regresión Polinomial:", mae_regresion_polinomial)
```

*Revisar script para visualizar los demas modelos propuestos.

LINK:

Resultados de los Modelos:

Modelo	MSE	R^2	MAE
Regresión Lineal	6449.70	0.0102	61.02
Regresión Polinomial	6550.85	-0.0053	61.42
Regresión de Árboles de Decisión	9955.30	-0.5277	71.97
Regresión de Bosques Aleatorios	8155.98	-0.2516	66.86
Red Neuronal	6449.22	0.0103	-

Conclusión:

- Los modelos tradicionales de regresión no son adecuados para capturar la complejidad de los datos: Los modelos de regresión lineal, polinomial y de árboles de decisión mostraron un desempeño limitado en la predicción del gasto total. Esto sugiere que la relación entre las variables independientes y la variable dependiente puede ser no lineal o compleja, lo que requiere modelos más sofisticados para su captura.

- Se necesitan técnicas avanzadas de modelado para mejorar la predicción: A pesar de intentar varios modelos, incluidos bosques aleatorios y redes neuronales, no pudimos lograr una mejora significativa en la capacidad predictiva. Esto sugiere la necesidad de explorar técnicas más avanzadas de modelado, como modelos de aprendizaje profundo o ensamblados, para capturar mejor las relaciones complejas entre las variables.

- El análisis detallado de los datos y la ingeniería de características pueden ser clave: La calidad de los modelos depende en gran medida de la calidad de los datos y la selección de características relevantes. Explorar más a fondo las relaciones entre las variables, identificar nuevas características importantes y preprocesar los datos de manera adecuada podrían ser pasos críticos para mejorar el desempeño del modelo.