Desglose detallado del código de predicción de presión arterial sistólica

1. Importación de librerías

```
import pandas as pd
import numpy as np
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.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
```

¿Qué hace cada una?

- `pandas`: Manipulación y análisis de datos, principalmente con DataFrames.
- `numpy`: Operaciones matemáticas eficientes con arrays.
- `matplotlib.pyplot`: Gráficos básicos en Python.
- `seaborn`: Visualizaciones estadísticas avanzadas y estéticas.
- `sklearn.model_selection.train_test_split`: División de datos en entrenamiento y prueba.
- `sklearn.linear_model.LinearRegression`: Implementación de regresión lineal.
- `sklearn.metrics.mean_squared_error`: Cálculo del error cuadrático medio.
- `sklearn.metrics.r2_score`: Cálculo del coeficiente de determinación R^2.
- `statsmodels.api.sm`: Modelado estadístico avanzado.

---

2. Configuración visual y semilla aleatoria

```
plt.style.use('seaborn-v0_8')
np.random.seed(42)
```

- `plt.style.use('seaborn-v0_8')`: Aplica el estilo de visualización de Seaborn a matplotlib.
- `np.random.seed(42)`: Fija una semilla para reproducibilidad de resultados aleatorios.

---

3. Generación de datos simulados

```
def generar_datos_pa(n=300):
    edad = np.random.randint(18, 81, size=n)
    imc = np.random.normal(27, 4, size=n)
    imc = np.clip(imc, 18, 38)
    colesterol = np.random.normal(200, 35, size=n)
    colesterol = np.clip(colesterol, 130, 300)
    pa_sistolica = 100 + 0.6 * edad + 0.8 * imc + 0.15 * colesterol +
np.random.normal(0, 8, size=n)
    return pd.DataFrame({'edad': edad, 'imc': imc, 'colesterol': colesterol,
'pa_sistolica': pa_sistolica})
```

- `np.random.randint(18, 81, size=n)`: Genera edades entre 18 y 80.
- `np.random.normal(27, 4, size=n)`: Genera valores de IMC con media 27 y desviación estándar 4.
- `np.clip(imc, 18, 38)`: Restringe el IMC entre 18 y 38.
- `np.random.normal(200, 35, size=n)`: Niveles de colesterol con media 200 y desviación estándar 35.
- `np.clip(colesterol, 130, 300)`: Restringe el colesterol entre 130 y 300.
- `pa_sistolica`: Calcula la presión arterial sistólica con un modelo lineal y añade ruido.

---

4. Exploración de datos

```
df = generar_datos_pa(300)
print(df.head())
print(df.describe())
```

- `df.head()`: Muestra las primeras filas.
- `df.describe()`: Estadísticas descriptivas de las columnas.

---

5. Visualización de distribuciones

```
plt.figure(figsize=(12, 10))
for i, col in enumerate(['edad', 'imc', 'colesterol', 'pa_sistolica'], 1):
    plt.subplot(2, 2, i)
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribución de {col}')
plt.tight_layout()
```

```
plt.savefig('distribucion_variables_pa.png')
```

- `plt.subplot(2, 2, i)`: Organiza 4 gráficos en una cuadrícula de 2x2.
- `sns.histplot(df[col], kde=True)`: Histograma con curva de densidad.

---

6. Matriz de correlación

```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlaciones entre variables')
plt.savefig('correlacion_pa.png')
```

- `df.corr()`: Matriz de correlación.
- `sns.heatmap()`: Mapa de calor para visualizar correlaciones.

---

7. Regresión lineal simple

```
for predictor in ['edad', 'imc', 'colesterol']:
    X = df[predictor].values.reshape(-1, 1)
    y = df['pa_sistolica'].values
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
    modelo = LinearRegression()
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
```

- `X = df[predictor].values.reshape(-1, 1)`: Convierte la columna en matriz 2D.
- `train_test_split()`: Divide en conjuntos de entrenamiento y prueba.
- `LinearRegression().fit()`: Entrena el modelo.
- `predict()`: Predice en el conjunto de prueba.
- `mean_squared_error()`: Error cuadrático medio.
- `r2_score()`: R^2.

---

8. Regresión lineal múltiple

```
X_multi = df[['edad', 'imc', 'colesterol']]
y = df['pa_sistolica']
modelo_multi = LinearRegression()
modelo_multi.fit(X_train, y_train)
y_pred_multi = modelo_multi.predict(X_test)
```

- `df[['edad', 'imc', 'colesterol']]`: Selección de múltiples columnas.
- `LinearRegression().fit()`: Entrena el modelo múltiple.

---

9. Predicción para nuevos pacientes

```
def predecir_pa_sistolica(edad, imc, colesterol, modelo=modelo_multi):
    paciente = pd.DataFrame([[edad, imc, colesterol]], columns=['edad', 'imc',
'colesterol'])
    pa_predicha = modelo.predict(paciente)[0]
    return pa_predicha
```

- `pd.DataFrame([[edad, imc, colesterol]])`: Crea DataFrame con datos del paciente.
- `predict(paciente)[0]`: Predice la PA sistólica.

Este es el desglose paso a paso del código. ¿Quieres que nos enfoquemos en alguna parte en particular o que profundice en algo más?