

✓ Regresión simple y Múltiple con Python

Python en el Ámbito Científico

Sesión 3

Alan Badillo Salas (alan@nomadacode.com)

✓ Repaso - Estadística descriptiva y la visualización de datos en Python

En la [Sesión 3](#) hemos trabajado la visualización de datos y la estadística descriptiva con Python.

Principalmente hemos desarrollado la capacidad de determinar la estructura de la información contenida en un conjunto de datos o *dataset*.

Un *dataset* está configurado por ejes de datos que naturalmente se asocian a las columnas y registros que son asociados a las filas.

Por ejemplo, para un fenómeno real que involucra el ingreso familiar podríamos tener un *dataset* como el siguiente:

Tipo	Integrantes	Padre	Madre	Hijos	Familiares	Otros	Ingreso Padre	Ingreso Madre	Ingreso Familia
Familia Completa	5	1	1	3	0	0	15000	8000	-
Familia Externa	7	0	1	4	2	0	-	12000	9000
Pareja Sin Hijos	2	1	1	0	0	0	6000	7000	-
Pareja Con Hijos	3	1	1	1	0	0	10000	9000	-
Familia Completa	4	1	1	2	0	0	10000	0	-
Familia Completa	5	1	1	3	0	0	0	8000	-
Familia Extendida	6	1	1	3	1	0	22000	15000	0
Mixto	3	0	0	0	0	3	-	-	-
Pareja Atípica	4	1	1	1	0	1	8000	0	-

En el análisis de datos debemos inspeccionar la estructura y organización de los datos.

Podemos usar la jerarquía de análisis clásica:

- **Naturaleza:** Cuantitativa (Numérica) o Cualitativa (Categórica)

- **Cuantitativa (Numérica):** Continua o Discreta
 - **Continua:** Temporal, Espacial, Cantidad, Fuerza o Probabilidad
 - Positiva y/o Negativa
 - Acotada o Infinita
 - **Discreta:** Temporal, Espacial, Cantidad, Potencia o Binaria
 - Positiva y/o Negativa
 - Finita o Infinita
- **Cualitativa (Categorica):** Singular, Binaria o Multiclase
 - **Singular:** Ordenada o Nominal (Sin orden)
 - **Binaria:** Ordenada o Nominal (Sin orden)
 - Contraste Positivo o Negativo
 - Probabilidad exacta
 - Relación de Existencia
 - Relación Pertenencia
 - **Multiclase:** Ordenada o Nominal (Sin orden)
 - Con orden Positivo y/o Negativo
- **Binaria:** Unifica la naturaleza cuantitativa discreta binaria con la naturaleza cualitativa binaria
 - **Existencia:** Proviene de una relación de existencia
 - Singular
 - Binaria
 - Multiclase
 - **Pertenencia:** Proviene de una relación de pertenencia
 - Única
 - Múltiple
 - **Probabilidad:** Proviene de una probabilidad absoluta
 - Directa
 - Dual
 - Indirecta
 - **Contraste:** Proviene del contraste de dos categorías

- Afirmativa
- Negativa
- **Activación:** Proviene de un evento o alerta
 - Encendido
 - Apagado
 - Alto
 - Bajo
 - Activar
 - Desactivar
 - ...
- **Segmentación:** Proviene de un grupo o partición
 - Grupo 1
 - Grupo 2
 - ...

Por ejemplo, para el **dataset** de ingreso familiar tenemos:

Eje de datos	Naturaleza	Tipo	Subtipo	Modelo	Responde
Tipo	Cualitativa (Categórica)	Multiclase	Nominal	-	Tipo de Hogar
Integrantes	Cuantitativa (Numérica)	Discreta	Cantidad	Positiva / Infinita	Número de integrantes
Padre	Binaria	Existencia	Singular	-	Hay padre
Madre	Binaria	Existencia	Singular	-	Hay madre
Hijos	Cuantitativa (Numérica)	Discreta	Cantidad	Positiva / Infinita	Número de de Hijos
Familiares	Cuantitativa (Numérica)	Discreta	Cantidad	Positiva / Infinita	Número de familiares
Otros	Cuantitativa (Numérica)	Discreta	Cantidad	Positiva / Infinita	Número de amigos o externos
Ingreso Padre	Cuantitativa (Numérica)	Continua	Cantidad	Positiva / Infinita	Dinero que aporta el padre al hogar
Ingreso Madre	Cuantitativa (Numérica)	Continua	Cantidad	Positiva / Infinita	Dinero que aporta la madre al hogar
Ingreso Familiares	Cuantitativa (Numérica)	Continua	Cantidad	Positiva / Infinita	Dinero que aportan otros familiares al
Ingreso Otros	Cuantitativa (Numérica)	Continua	Cantidad	Positiva / Infinita	Dinero que aportan otros al hogar

Una vez identificada la naturaleza de los datos, se puede hacer un tratamiento de los datos que consiste en:

- **Limpieza (Corrección de Datos)** - Busca que todos los valores sean correctos, por ejemplo, quitar valores nulos (filtrar los datos), rellenar los datos faltantes (simular los datos), corregir los datos mal estructurados como convertir los ceros a valores nulos si se da el caso o normalizar los datos mal formados como adecuarlos a las categorías únicas
- **Estructuración (Transformación / Preprocesamiento de Datos)** - Busca que todos los

datos se puedan analizar correctamente, por ejemplo, los ejes de datos categóricos convertirlos en ejes **One-Hot** binarios, por ejemplo, si es hombre, mujer u otro separarlo en el eje binario *hombre*, el eje binario *mujer* y el eje binario *otro*, los ejes discretos separarlos en grupos binarios, por ejemplo, agrupar las edades de 5 en 5 años, generando los grupos *edad* : 0 – 4, *edad* : 5 – 9, *edad* : 10 – 14, . . . y los ejes que codifiquen múltiples valores separarlos, por ejemplo, una coordenada (*latitud*, *longitud*) convertirla en un eje de *latitud* y un eje de *longitud*

- **Procesamiento (Modelo de Datos)** - Busca procesar los datos mediante un modelo de análisis de los datos que genere nuevos datos o una síntesis sobre las observaciones. Por ejemplo, un *Modelo de Machine Learning* (Modelo de Aprendizaje Automático) puede buscar una clasificación o regresión de algún eje de datos respecto a los otros ejes de datos, un *Modelo de Deep Learning* (Modelo de Aprendizaje Profundo) puede buscar una clasificación o regresión usando redes neuronales, un *Modelo Matemático o Estadístico* puede buscar describir las probabilidades o relaciones entre los datos, realizar alguna prueba de hipótesis y determinar los estadísticos estimados.
- **Visualización (Esquema de Datos)** - Busca explicar los ejes de datos mediante gráficas y esquemas visuales que expliquen el comportamiento de los datos para guiar el análisis de los datos.
- **Automatización (Aplicación de Datos)** - Busca sintetizar los puntos anteriores para producir una aplicación en tiempo real que de respuesta automática a los datos, por ejemplo, dashboards de métricas en tiempo real, aplicaciones de alertas en tiempo real, chatbots y generadores de información automática, reportes automatizados, etc.

Entonces, para lograr un correcto análisis del **dataset** se necesita:

- **Adquirir los datos y manipularlos** - Mediante herramientas como pandas, numpy, pillow, entre otras se pueden cargar *datasets*, imágenes, audio, video y demás, para manipularlo y transformarlo, por ejemplo, seleccionar columnas, filtrar filas, agrupar datos, mezclar datos, etc.
- **Visualizar los datos y reportarlos** - Mediante herramientas como pandas, numpy, scipy, scikit-learn, entre otras, se pueden visualizar ejes de datos en forma de columnas (*series* o arreglos) o tablas (*dataframes* o matrices) para mostrar la distribución de los datos y su comportamiento espacio-temporal y estadístico.

Dependiendo la naturaleza de los datos, aplicaremos diferentes tipos de visualización:

- **Eje numérico continuo:** Representa muchos valores decimales de la misma naturaleza, por ejemplo, pesos, estaturas, longitudes, precios o temperaturas

- **Histograma** - Para conocer la distribución de los datos
 - Gráfica de Histograma (histplot)
 - Gráfica de Densidad (kdeplot)
 - Gráfica de Caja Clásica (boxplot)
 - Gráfica de Cajas Múltiples (boxenplot)
 - Gráfica de Densidad Simétrica (violinplot)
- **Eje numérico discreto:** Representa muchos valores enteros de la misma naturaleza, por ejemplo, edad, número de piezas, número de integrantes, número de horas o día de observación, estos pueden indicar valores simples como la edad de una persona o acumulados como el número de hijos
 - **Histograma** - Para conocer la distribución de los datos
 - Gráfica de Histograma (histplot)
 - Gráfica de Caja Clásica (boxplot)
 - Gráfica de Cajas Múltiples (boxenplot)
 - Gráfica de Densidad Simétrica (violinplot)
 - **Barras** - Para conocer el número de elementos o su probabilidad para cada valor (si es finito)
 - Gráfica de Conteos (countplot)
 - Gráfica de Probabilidad (countplot / probability)
- **Eje categórico multiclase:** Representa muchos valores de diferentes categorías de la misma naturaleza, por ejemplo, tipo de empleo (desempleado, gubernamental, empresa, campo, doméstico), escolaridad (primaria, secundaria, preparatoria, graduado) o producto (coca-cola, gansito, ...)
 - **Histograma** - Para conocer la distribución de los datos
 - Gráfica de Densidad Simétrica (violinplot)
 - **Barras** - Para conocer el número de elementos o su probabilidad para cada categoría
 - Gráfica de Conteos (countplot)
 - Gráfica de Probabilidad (countplot / probability)
 - **Pastel** - Para conocer la proporción de cada categoría
 - Gráfica de Pastel (pie)
 - Gráfica de Dona (pie / artist)

- **Eje numérico continuo vs numérico continuo:** Representa el contraste entre dos ejes continuos, por ejemplo, peso vs estatura (persona), peso vs precio (producto) o superficie vs precio (terreno)
 - **Correlación** - Muestra la correlación entre los datos
 - Gráfica de Puntos (scatterplot)
 - Gráfica de Densidad (kdeplot / xy)
 - Gráfica de Unión (jointplot / scatter, reg, resid, kde, hist, hex)
- **Eje numérico continuo vs numérico discreto:** Representa el contraste entre un eje continuo y un eje discreto, por ejemplo, peso vs edad (persona), valoración vs precio (producto) o número de baños vs precio (terreno)
 - **Histograma** - Para conocer la distribución de los datos segmentados por cada valor discreto
 - Gráfica de Histograma (histplot)
 - Gráfica de Densidad (kdeplot)
 - Gráfica de Caja Clásica (boxplot)
 - Gráfica de Cajas Múltiples (boxenplot)
 - Gráfica de Densidad Simétrica (violinplot)
 - **Correlación** - Muestra la correlación entre los datos
 - Gráfica de Puntos (scatterplot)
 - Gráfica de Densidad (kdeplot / xy)
 - Gráfica de Unión (jointplot / scatter, reg, resid, kde, hist, hex)
- **Eje numérico discreto vs numérico discreto:** Representa el contraste entre un eje discreto y un eje discreto, por ejemplo, peso vs edad (persona), valoración vs precio (producto) o número de baños vs precio (terreno)
 - **Barras** - Para conocer el número de elementos o su probabilidad para cada valor (si es finito) del eje principal por cada valor discreto del eje secundario
 - Gráfica de Conteos (countplot)
 - Gráfica de Probabilidad (countplot / probability)
- **Eje numérico continuo vs categórico multiclase:** Representa el contraste entre un eje continuo y un eje categórico, por ejemplo, peso vs sexo (hombre, mujer, ...) (persona), departamento (lácteos, carnes, ...) vs precio (producto) o zona (urbana, rural, ...) vs precio (terreno)

- **Histograma** - Para conocer la distribución de los datos segmentados por cada categoría
 - Gráfica de Histograma (histplot)
 - Gráfica de Densidad (kdeplot)
 - Gráfica de Caja Clásica (boxplot)
 - Gráfica de Cajas Múltiples (boxenplot)
 - Gráfica de Densidad Simétrica (violinplot)

También existen gráficos mixtos que podrán codificar múltiples ejes numéricos y categóricos.

Para más tipos de visualizaciones nos podemos guiar mediante:

<https://www.data-to-viz.com>

✓ Ejemplo - Análisis y visualización de la Diabetes

Column Name	Description
Age	Age of the person
Sex	Gender of the person
Ethnicity	Ethnic background of the person
BMI	Body Mass Index, a measure of body fat based on height and weight
Waist_Circumference	Measurement of waist circumference
Fasting_Blood_Glucose	Blood glucose level after a period of fasting
HbA1c	Hemoglobin A1c, a measure of average blood sugar levels over the past 2-3 months
Blood_Pressure_Systolic	Systolic blood pressure reading
Blood_Pressure_Diastolic	Diastolic blood pressure reading
Cholesterol_Total	Total cholesterol level in the blood
Cholesterol_HDL	High-density lipoprotein (good cholesterol) level in the blood
Cholesterol_LDL	Low-density lipoprotein (bad cholesterol) level in the blood
GGT	Gamma-glutamyl transferase, a liver enzyme indicative of liver health
Serum_Urate	Uric acid concentration in the blood
Physical_Activity_Level	Self-reported level of physical activity
Dietary_Intake_Calories	Estimated daily caloric intake
Alcohol_Consumption	Alcohol consumption status (e.g., none, moderate, high)
Smoking_Status	Smoking status (e.g., current, former, never)
Family_History_of_Diabetes	Indicates whether the person has a family history of diabetes (1 = yes, 0 = no)
Previous_Gestational_Diabetes	History of gestational diabetes (1 = yes, 0 = no)


▼ Obtener los datos

```
import pandas

url = "https://github.com/dragonnomada/ipn-cic-pycien-abril-2025/"
url += "raw/refs/heads/main/"
url += "datasets/diabetes.csv"

diabetes = pandas.read_csv(url)

diabetes.head()
```



	Unnamed: 0	Age	Sex	Ethnicity	BMI	Waist_Circumference	Fasting_Blood_Glu
0	0	58	Female	White	35.8	83.4	
1	1	48	Male	Asian	24.1	71.4	
2	2	34	Female	Black	25.0	113.8	
3	3	62	Male	Asian	32.7	100.4	
4	4	27	Female	Asian	33.5	110.8	

5 rows x 21 columns

▼ Mostrar la información general


```
diabetes.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 21 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Unnamed: 0                                10000 non-null  int64
1   Age                                         10000 non-null  int64
2   Sex                                         10000 non-null  object
3   Ethnicity                                  10000 non-null  object
4   BMI                                         10000 non-null  float64
5   Waist_Circumference                       10000 non-null  float64
6   Fasting_Blood_Glucose                     10000 non-null  float64
7   HbA1c                                       10000 non-null  float64
8   Blood_Pressure_Systolic                    10000 non-null  int64
9   Blood_Pressure_Diastolic                   10000 non-null  int64
10  Cholesterol_Total                           10000 non-null  float64
11  Cholesterol_HDL                             10000 non-null  float64
12  Cholesterol_LDL                             10000 non-null  float64
13  GGT                                          10000 non-null  float64
14  Serum_Urate                                10000 non-null  float64
15  Physical_Activity_Level                     10000 non-null  object
16  Dietary_Intake_Calories                     10000 non-null  int64
17  Alcohol_Consumption                         6680 non-null   object
18  Smoking_Status                             10000 non-null  object
19  Family_History_of_Diabetes                  10000 non-null  int64
20  Previous_Gestational_Diabetes               10000 non-null  int64
dtypes: float64(9), int64(7), object(5)
memory usage: 1.6+ MB
```

✓ Mostrar la primera fila

```
diabetes.loc[0]
```



0

Unnamed: 0	0
Age	58
Sex	Female
Ethnicity	White
BMI	35.8
Waist_Circumference	83.4
Fasting_Blood_Glucose	123.9
HbA1c	10.9
Blood_Pressure_Systolic	152
Blood_Pressure_Diastolic	114
Cholesterol_Total	197.8
Cholesterol_HDL	50.2
Cholesterol_LDL	99.2
GGT	37.5
Serum_Urate	7.2
Physical_Activity_Level	Moderate
Dietary_Intake_Calories	1538
Alcohol_Consumption	Moderate
Smoking_Status	Never
Family_History_of_Diabetes	0
Previous_Gestational_Diabetes	1

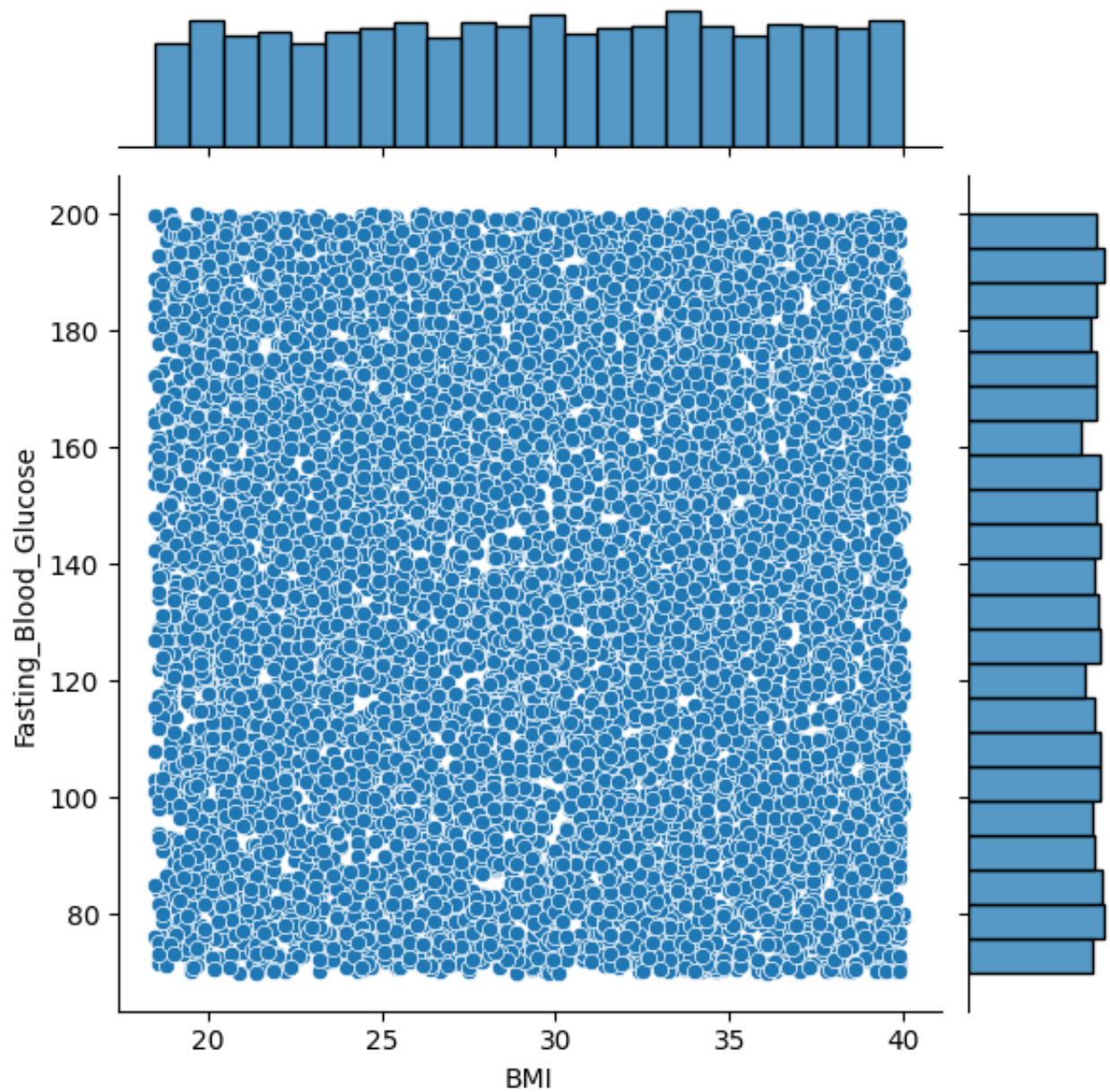
dtype: object

✓ IMC (Continua) vs Glucosa (Continua)

```
import seaborn
```

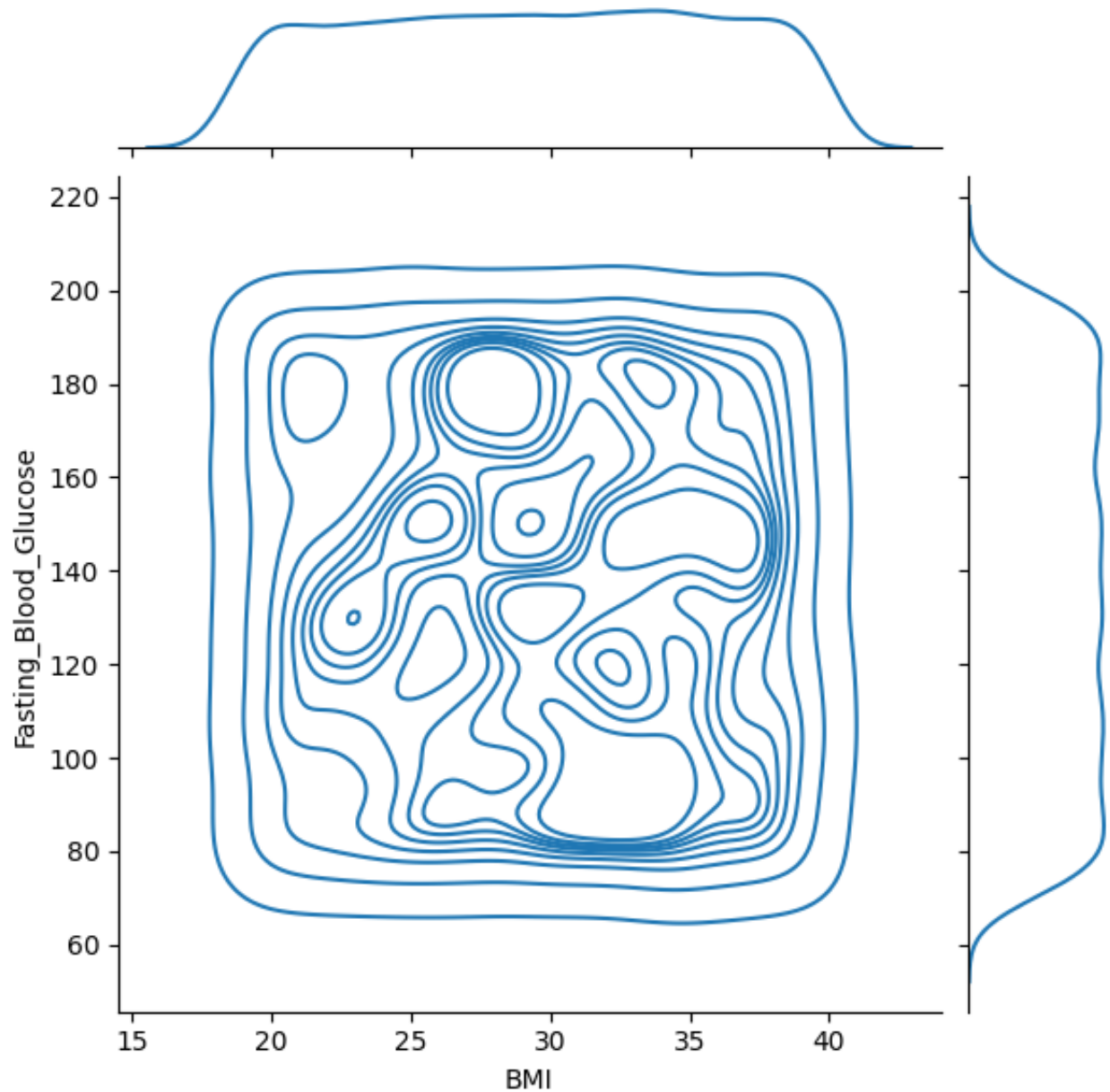
```
seaborn.jointplot(diabetes, x="BMI", y="Fasting_Blood_Glucose")
```

```
<seaborn.axisgrid.JointGrid at 0x7a51aa6a9750>
```



```
seaborn.jointplot(diabetes, x="BMI", y="Fasting_Blood_Glucose", kind="kde")
```

```
>>> <seaborn.axisgrid.JointGrid at 0x7a51f83cb810>
```



✓ Filtrar datos de hipótesis

```
diabetes["Alcohol_Consumption"].unique()
```

```
>>> array(['Moderate', 'Heavy', nan], dtype=object)
```

```
diabetes["Smoking_Status"].unique()
```

```
>>> array(['Never', 'Current', 'Former'], dtype=object)
```

```
diabetes["Physical_Activity_Level"].unique()
```

```
⇒ array(['Moderate', 'Low', 'High'], dtype=object)
```

```
diabetes["Ethnicity"].unique()
```

```
⇒ array(['White', 'Asian', 'Black', 'Hispanic'], dtype=object)
```

```
diabetes["Sex"].unique()
```

```
⇒ array(['Female', 'Male'], dtype=object)
```

```
diabetes1 = diabetes[
    (diabetes["Sex"] == "Male") &
    (diabetes["Ethnicity"] == "Hispanic") &
    (diabetes["Physical_Activity_Level"] == "Low") &
    (diabetes["Alcohol_Consumption"] == "Heavy") &
    (diabetes["Smoking_Status"] == "Former")
]

diabetes1.head()
```


```
⇒
```

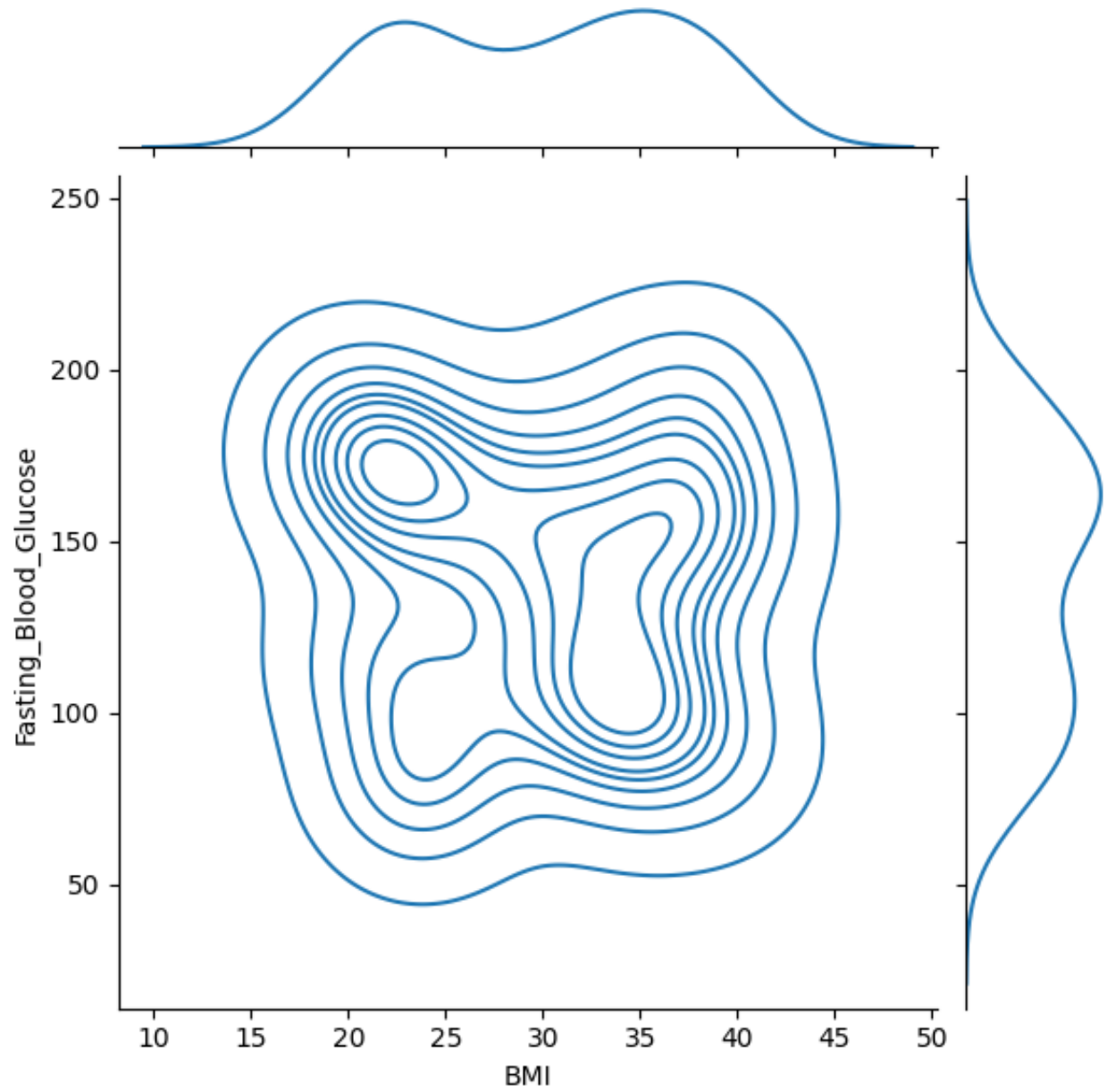
	Unnamed: 0	Age	Sex	Ethnicity	BMI	Waist_Circumference	Fasting_Blood_Glu
314	314	54	Male	Hispanic	36.6		92.5
320	320	52	Male	Hispanic	22.7		87.5
363	363	60	Male	Hispanic	21.7		82.6
707	707	40	Male	Hispanic	34.0		88.8
764	764	44	Male	Hispanic	20.7		108.7

5 rows x 21 columns


✓ IMC vs Glucosa en personas que fuman, beben y no hacen deporte

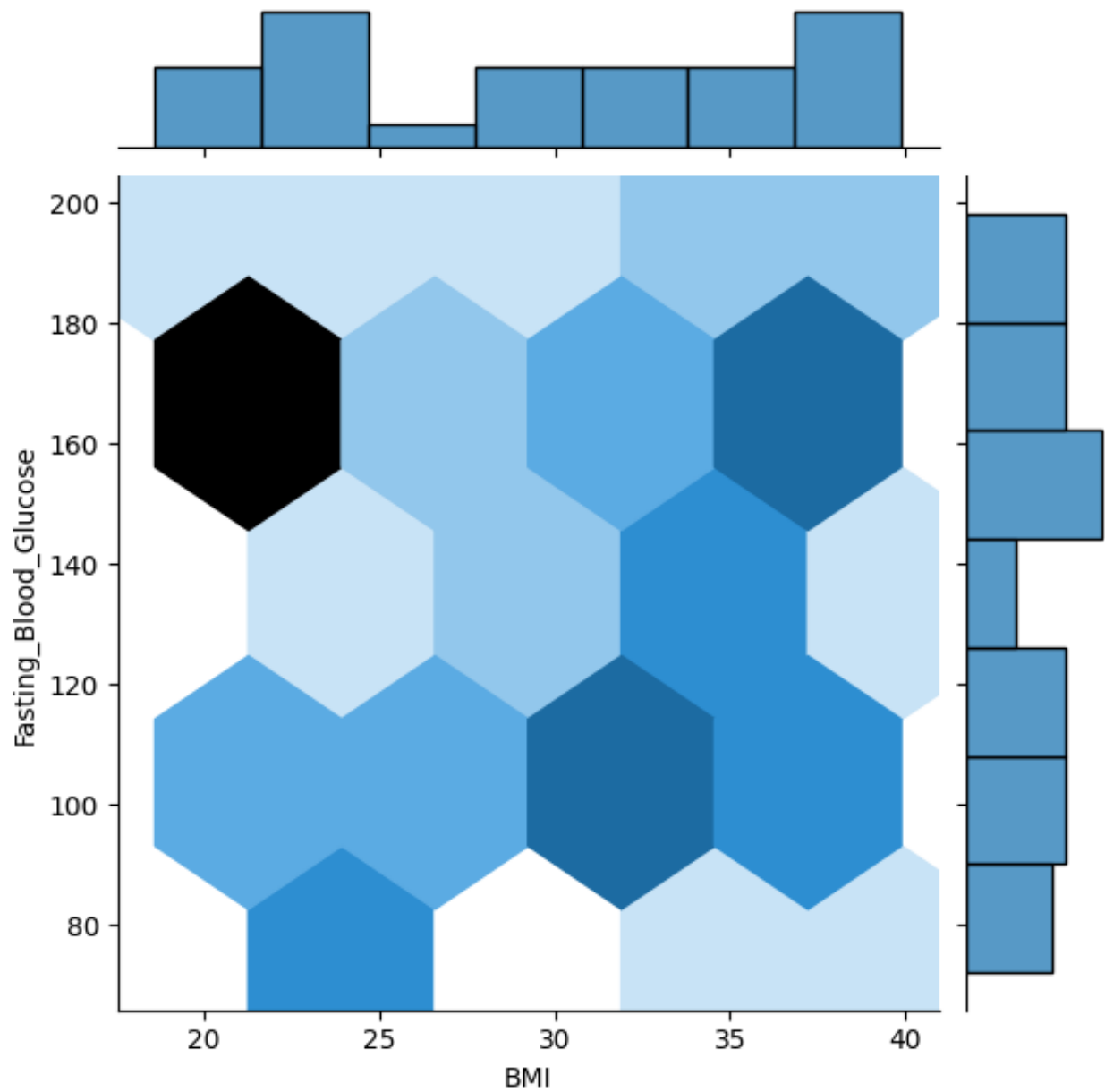
```
seaborn.jointplot(diabetes1, x="BMI", y="Fasting_Blood_Glucose", kind="kde")
```

 <seaborn.axisgrid.JointGrid at 0x7a51a805b990>




```
seaborn.jointplot(diabetes1, x="BMI", y="Fasting_Blood_Glucose", kind="hex")
```

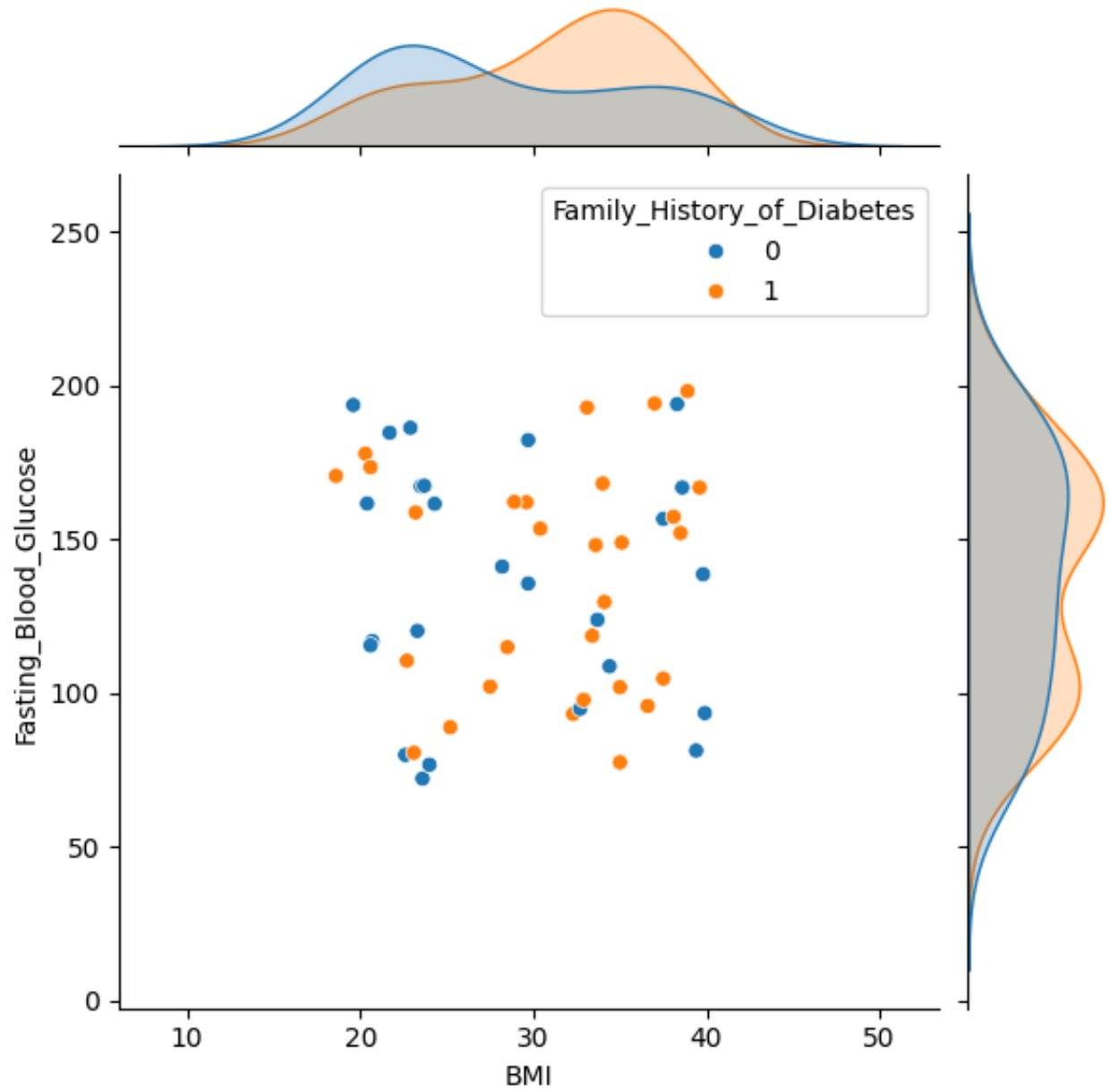
 <seaborn.axisgrid.JointGrid at 0x7a51aa6f1210>




✓ IMC vs Glucosa vs Historial Familiar de Diabetes

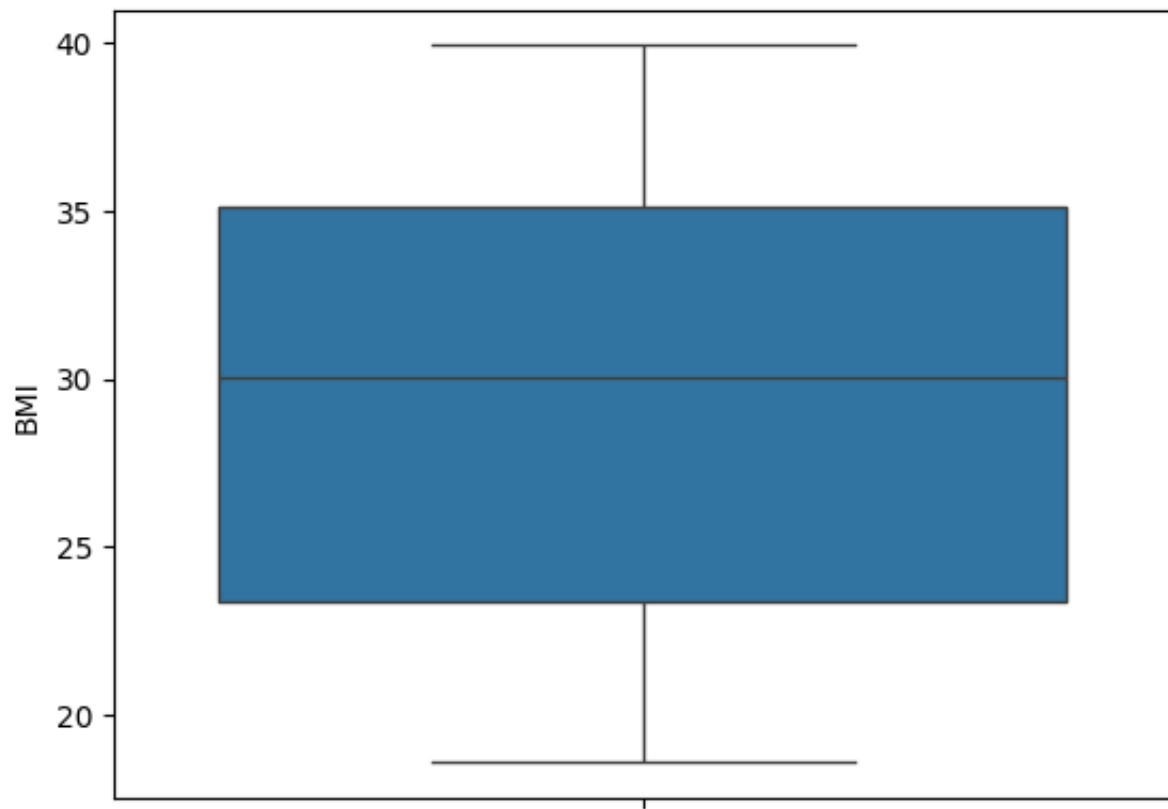
```
seaborn.jointplot(diabetes1, x="BMI", y="Fasting_Blood_Glucose", hue="Family_Hist
```

 <seaborn.axisgrid.JointGrid at 0x7a51a40f1210>



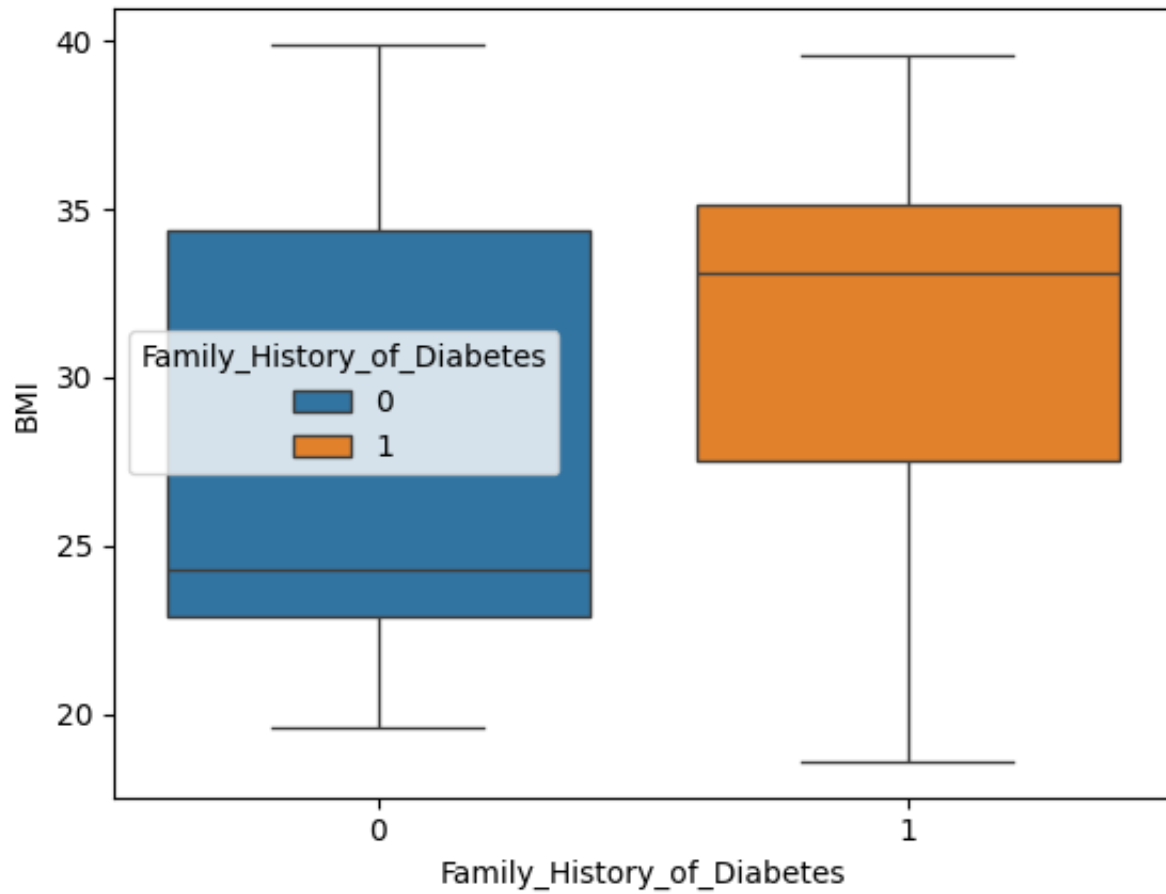

```
seaborn.boxplot(diabetes1, y="BMI")
```

 <Axes: ylabel='BMI'>



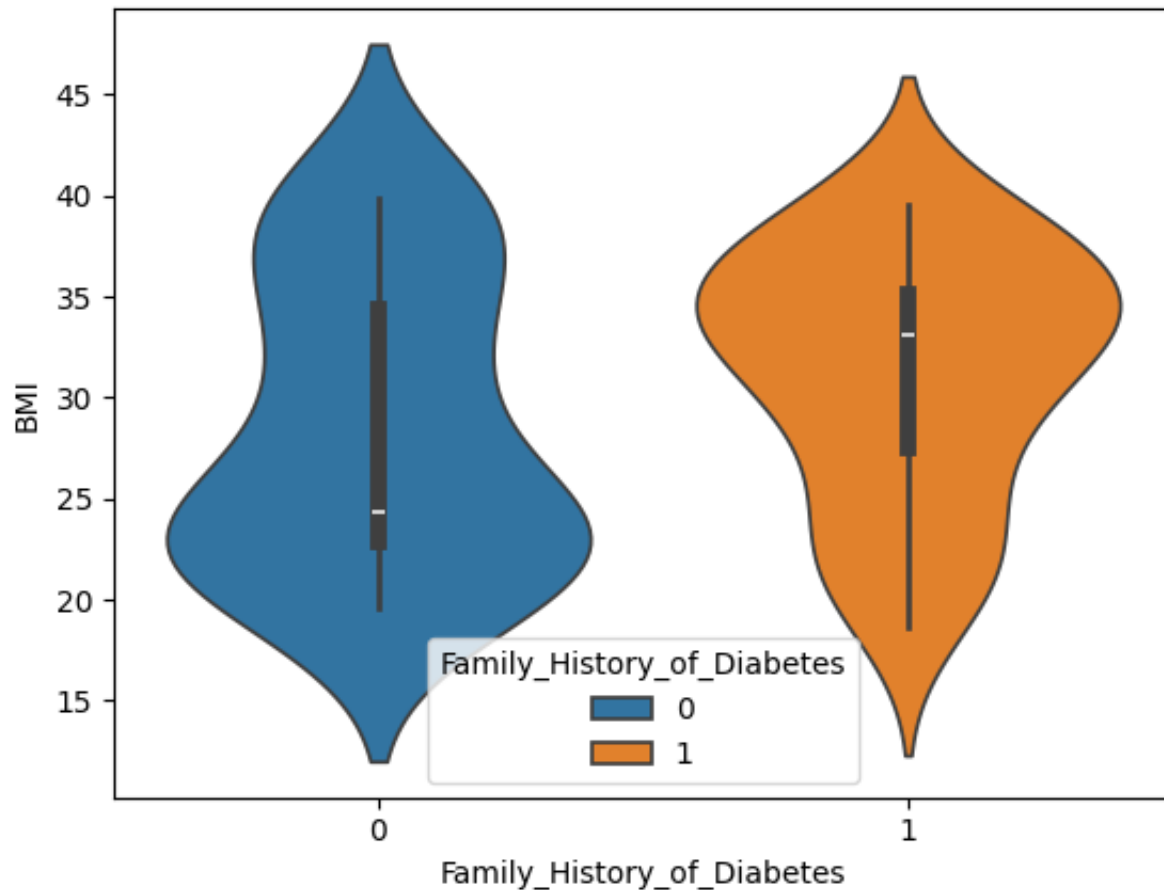
```
seaborn.boxplot(diabetes1, y="BMI", x="Family_History_of_Diabetes", hue = "Family.
```

 <Axes: xlabel='Family_History_of_Diabetes', ylabel='BMI'>



```
seaborn.violinplot(diabetes1, y="BMI", x="Family_History_of_Diabetes", hue = "Fam
```

```
<Axes: xlabel='Family_History_of_Diabetes', ylabel='BMI'>
```



✓ Revisión de la Tarea 2 y Tarea 2b

Revisión aquí

✓ Estadística Inferencial

Hasta ahora hemos trabajado con la Estadística Descriptiva, esta nos permite conocer y explorar los datos sin hacer suposiciones mayores.

Por ejemplo, para el *dataset* del *Titanic*, podemos entender cuales son los datos que tenemos y su naturaleza, por ejemplo, si es un eje numérico como la edad, entonces podemos entender cómo se distribuye la edad, o si el eje es categórico como el género, podemos contar cuántos datos hay por cada categoría.

Además, para ejes numéricos podemos analizar estadísticamente los datos principales que representan al eje, como su media y varianza, sus cuartiles y rango y demás.

Sin embargo, no es suficiente con poder describir los datos.

Sino que nuestro objetivo es construir un modelo capaz de predecir cuál será la respuesta de una variable en función de otras.

A este modelo lo conoceremos como **Modelo de Regresión**.

En términos generales el *Modelo de Regresión* busca explicar una variable de respuesta a partir de otras variables predictoras o *covariables*.

Existen dos tipos de *Modelos de Regresión*:

- **Modelo de Regresión Lineal**: Que ajusta un modelo lineal a los predictores.
- **Modelo de Regresión No-Lineal**: Que aumenta una función no lineal a los predictores.

Para entender un poco mejor el *Modelo de Regresión* es necesario pensar a la variable de respuesta y como el resultado de una combinación lineal o no-lineal de las variables predictoras x_1, x_2, \dots, x_k y parámetros desconocidos (coeficientes de regresión) $\beta_0, \beta_1, \dots, \beta_k$ tales que

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \dots + \beta_k \cdot x_k + \varepsilon$$

donde ε es un error normal que asumiremos entre los datos observados y el modelo de regresión.

A los coeficientes de regresión $\beta_0, \beta_1, \beta_2, \dots, \beta_k$ les llamaremos también los *parámetros* del modelo de regresión.

En particular, el parámetro desconocido β_0 se conoce como el *interceptor* o *bias* y su valor indica la respuesta constante cuándo todas las variables predictoras o covariables x_1, x_2, \dots, x_k son cero.

Si conociéramos los parámetros $\beta_0, \beta_1, \beta_2, \dots, \beta_k$, el modelo no se llamaría de regresión, solo sería un modelo de predicción y ya, pero al tener que descubrir quiénes son los $\beta_0, \beta_1, \beta_2, \dots, \beta_k$ que mejor explican los datos observados de y , entonces el modelo adquiere la forma de "regresión", ya que intenta recuperar los parámetros desconocidos a través de los datos.

Por ahora, solo debemos saber que existen varias formas de encontrar quiénes son los parámetros $\beta_0, \beta_1, \beta_2, \dots, \beta_k$ mediante regresión por mínimos cuadrados o máxima verosimilitud. Sin embargo, lo más importante ahora es entender cómo hacer la inferencia y la importancia de encontrar $\beta_0, \beta_1, \beta_2, \dots, \beta_k$.

✓ Inferencia y predicción

Para contextualizar el problema de la inferencia, vamos a pensar en un problema simple:

Imaginemos que tenemos el gasto familiar destinado a renta y el ingreso familiar total.

Entonces, el eje de datos `ingreso` será una variable numérica continua positiva con el monto total de dinero que aporta la familia al hogar.

Y, el eje de datos `gasto_renta` será una variable numérica continua positiva con el monto total de dinero que gasta la familia en la renta del hogar.

Por ejemplo, podemos tener los siguientes datos provenientes de una encuesta realizada a 20 hogares de familias que rentan en una zona conocida (por ejemplo, en la Condesa o Tepito)

Familia	Ingreso (\$)	Gasto Renta (\$)
1	23,450.00	7,276.00
2	18,730.00	5,156.72
3	35,600.00	11,232.24
4	42,100.00	13,357.00
5	27,900.00	7,934.70
6	31,400.00	9,577.66
7	16,850.00	4,974.15
8	24,700.00	7,410.00
9	21,300.00	6,232.56
10	19,800.00	5,441.40
11	38,200.00	11,076.00
12	28,300.00	8,347.86
13	25,500.00	6,936.00
14	30,000.00	9,600.00
15	22,100.00	6,418.90
16	17,650.00	5,294.15
17	26,800.00	8,040.00
18	33,300.00	10,095.30
19	29,400.00	8,341.20
20	36,700.00	10,636.36

En los datos observamos que algunas familias ganan más que otras y destinan montos mayores a la renta (caso idealista).

Entonces, podemos querer analizar los datos, partiendo por describir cómo se correlacionan los datos.

✓ Correlación entre y , x

La correlación es un procedimiento estándar para entender si hay una explicación entre las relaciones internas de dos ejes de datos.

Esto se logra a partir de la covarianza de los ejes y asumir que se comportan de forma normal.

Primero vamos inspeccionar los ejes de datos, tenemos dos ejes de datos con 20 muestras cada uno

```
x = [23450.00, 18730.00, 35600.00, 42100.00, 27900.00,  
     31400.00, 16850.00, 24700.00, 21300.00, 19800.00,  
     38200.00, 28300.00, 25500.00, 30000.00, 22100.00,  
     17650.00, 26800.00, 33300.00, 29400.00, 36700.00]
```

```
y = [7276.46, 5156.72, 11232.24, 13357.00, 7934.70,  
     9577.66, 4974.15, 7410.00, 6232.56, 5441.40,  
     11076.00, 8347.86, 6936.00, 9600.00, 6418.90,  
     5294.15, 8040.00, 10095.30, 8341.20, 10636.36]
```

Entonces, lo común es construir un *dataframe* que nos permita manipular los datos de una forma más elegante

```
import pandas

datos = pandas.DataFrame({
    "x": x,
    "y": y
})

datos.head()
```



	x	y
0	23450.0	7276.46
1	18730.0	5156.72
2	35600.0	11232.24
3	42100.0	13357.00
4	27900.0	7934.70



Próximos pasos:

[Ver gráficos recomendados](#)

[New interactive sheet](#)

Ahora podemos describir estadísticamente los datos

```
datos.describe()
```




	x	y
count	20.000000	20.000000
mean	27489.000000	8168.933000
std	7183.269963	2332.880573
min	16850.000000	4974.150000
25%	21900.000000	6372.315000
50%	27350.000000	7987.350000
75%	31875.000000	9723.825000
max	42100.000000	13357.000000

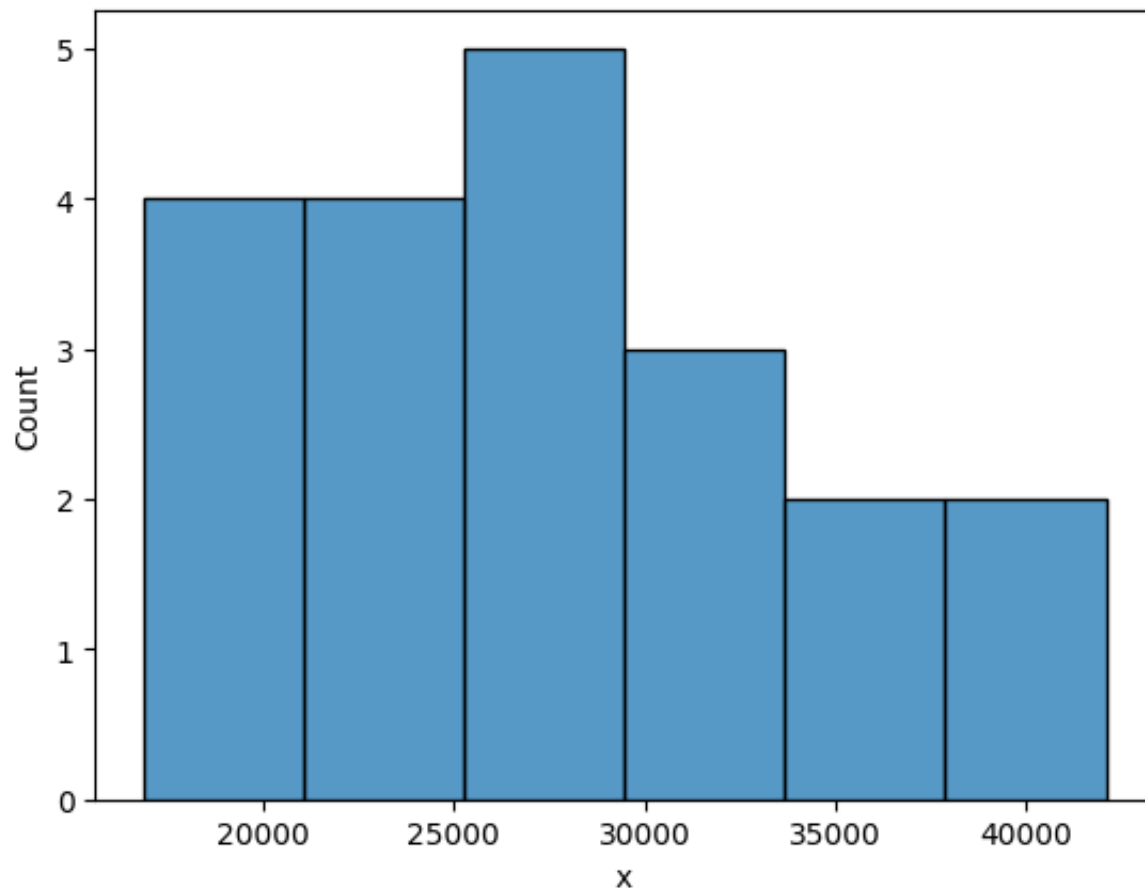


Observamos que el ingreso medio es de \$27, 489, mientras que el gasto promedio en renta es de \$8, 168.93.

Inspeccionemos la normalidad de los ejes observando sus histogramas


```
import seaborn  
  
seaborn.histplot(datos, x="x")
```

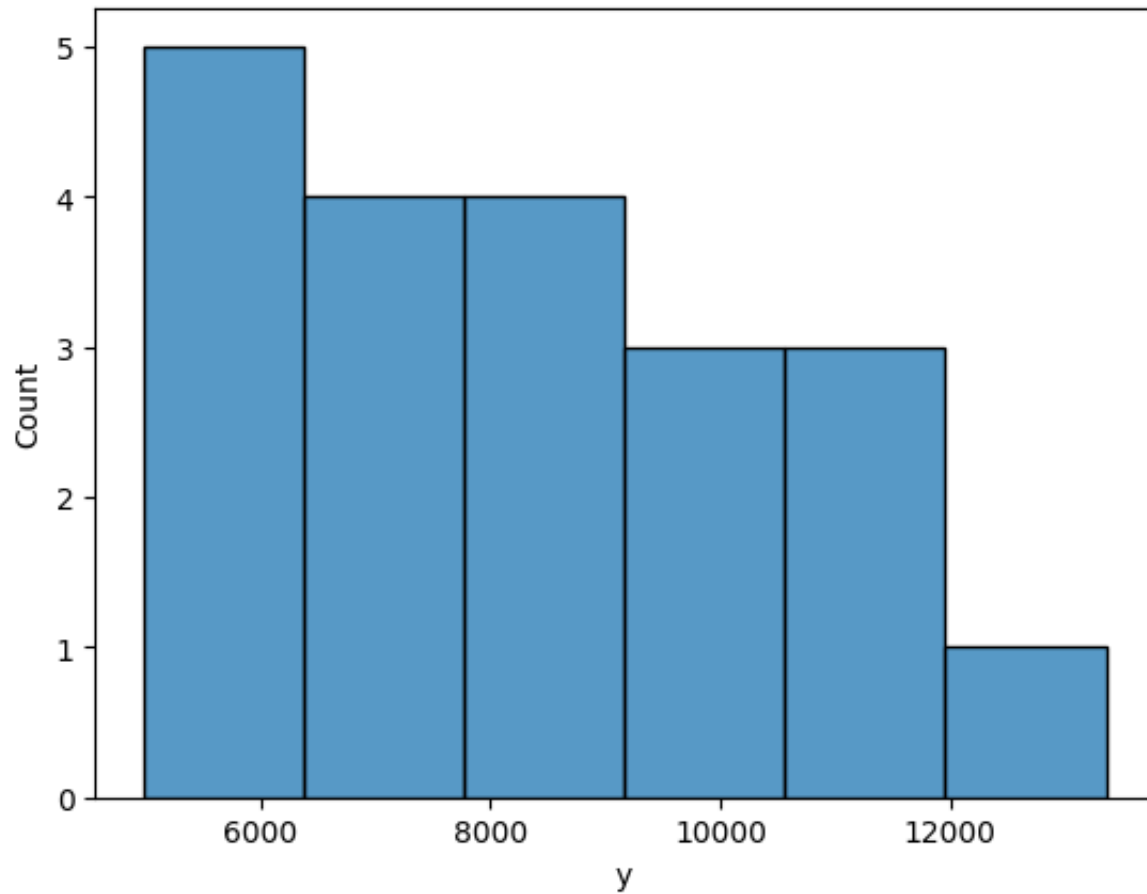
 <Axes: xlabel='x', ylabel='Count'>



Los ingresos parecen no ser muy normales, aunque hay cierta afinidad a serlo


```
seaborn.histplot(datos, x="y")
```

 <Axes: xlabel='y', ylabel='Count'>

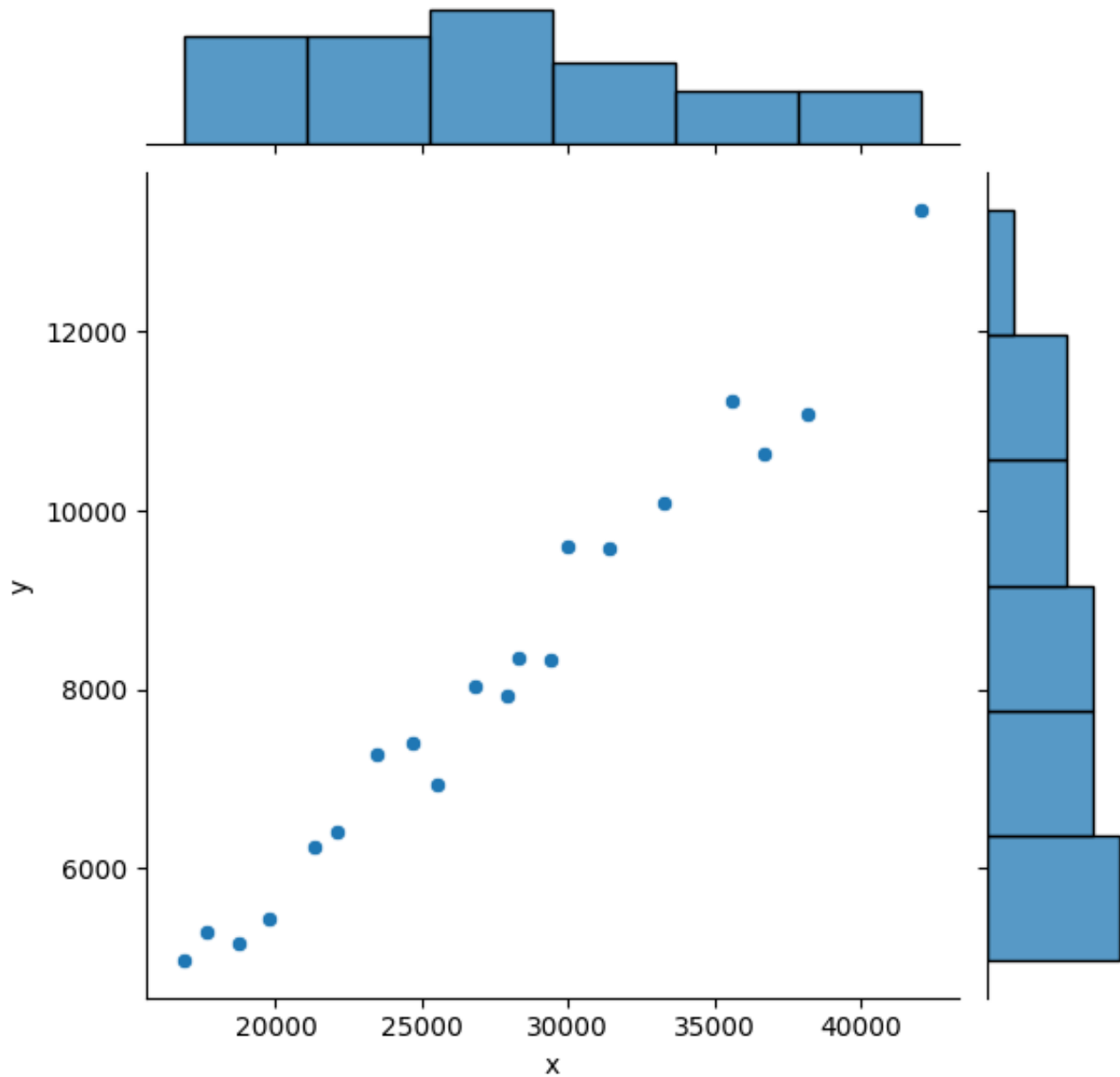


El gasto no parece comportarse de forma normal, parece sesgado y cargado a la izquierda

Ahora veamos los datos de ambas variables al mismo tiempo

```
seaborn.jointplot(datos, x="x", y="y")
```

```
<seaborn.axisgrid.JointGrid at 0x7a519fb54190>
```



Al parecer, la correlación entre la x (ingreso familiar) y la y (gasto en renta) están fuertemente relacionadas.

Una forma común de apreciar esto es mediante el indicador de Correlación de Pearson, o pruebas F.

Veamos la Correlación de Pearson entre ambos (que asume normalidad, cuidado)

```
datos["x"].corr(datos["y"])
```

```
np.float64(0.9877539196399292)
```

Vemos una fuerte correlación entre ambos ejes:

- **Correlación positiva cercana a 1** - Indica que ambos ejes crecen directamente, es decir, si el valor de un eje aumenta el otro también, por lo que **su correlación es fuerte y positiva**
- **Correlación negativa cercana a -1** - Indica que ambos ejes crecen inversamente, es decir, si el valor de un eje aumenta el otro disminuye, por lo que **su correlación es fuerte pero negativa**
- **Correlación cercana a 0** - Indica que ambos ejes no están correlacionados linealmente por factores normales, no significa que no se relacionen en otras formas no lineales, solo que directamente no lo hacen, por lo que **su correlación es nula**

En este caso, el valor cercano a +1 indica que la correlación es fuerte y positiva.

Esto implica que la variable predictora x puede explicar fácilmente a la variable de respuesta y

Es decir, el gasto de renta se puede explicar a través del ingreso familiar de forma efectiva.

Ahora veamos cómo lograr esta explicación

✓ Modelo de Regresión Lineal Simple

Sabiendo que tenemos una sola variable predictora x (una sola característica de información inicial o a priori), y una variable de respuesta y que se desea explicar a partir de la variable predictora x , entonces, podemos contruir un modelo lineal genérico que explique una variable a partir de la otra:

$$y = \beta_0 + \beta_1 \cdot x$$

Este será nuestro *Modelo de Regresión Lineal* dado que los parámetros son lineales y es *Simple* porque se basa en una única variable de predicción (una sola x).

Sin embargo, no tenemos idea de quiénes son β_0, β_1 , solo hacemos uso de ellos para generalizar un modelo común a todos los problemas x, y que se deseen predecir.

Pero, para nuestros datos específicos, debería haber una $\hat{\beta}_0, \hat{\beta}_1$ que de todas las posibles β_0, β_1 sean las mejores, es decir, las que mejor se ajusten a nuestros datos.

Para ejemplificar esto, vamos a proponer β_0 , β_1 con cualquier valor manual que pensemos que funcionaría, y ver cómo se comporta el modelo genérico en la predicción de las y dadas las x

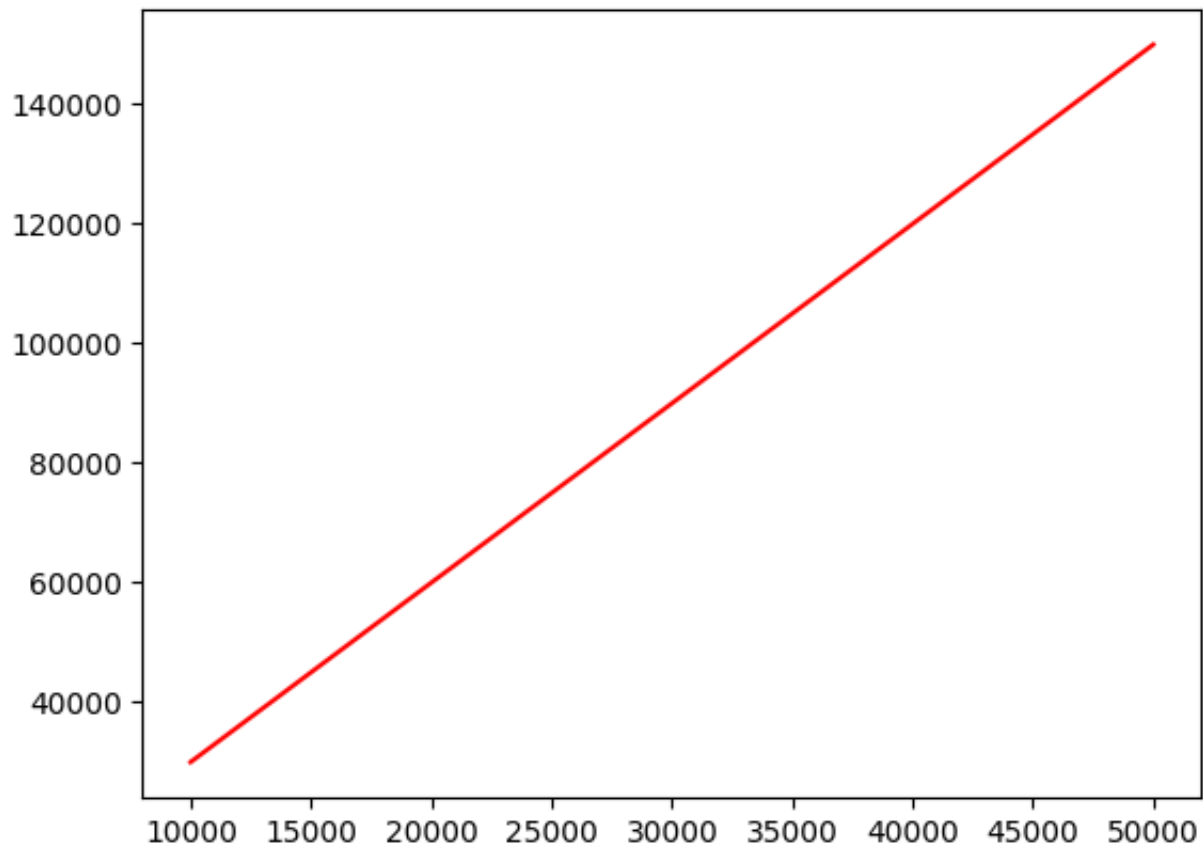
```
import matplotlib.pyplot as pyplot
import numpy

# 1. Supongamos que beta0 = 1, beta1 = 3
beta0 = 1
beta1 = 3

# 2. Para las x en [10000, 50000] predeciremos la y
# generamos 100 puntos distribuidos en el intervalo
x = numpy.linspace(10000, 50000, 100)

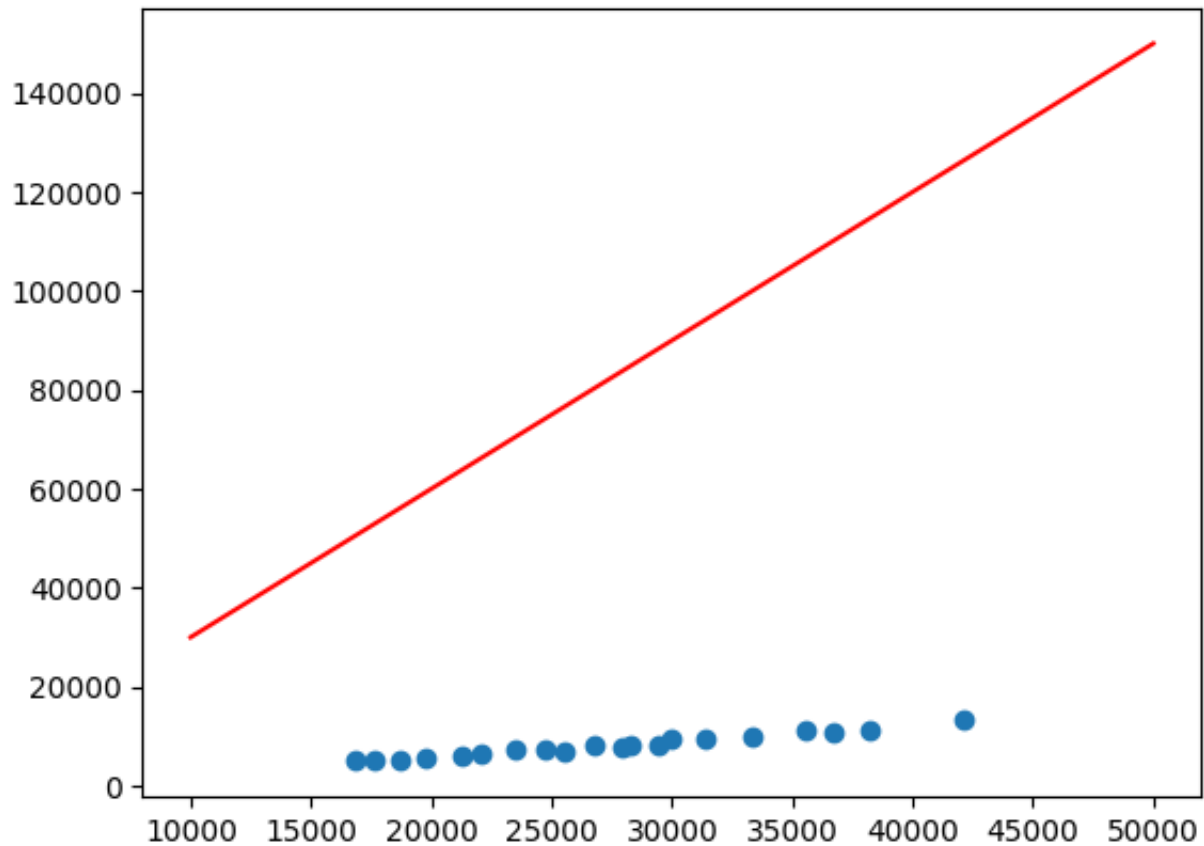
# 3. Entonces, para cada x podemos calcular quién es y
y = beta0 + beta1 * x

# 4. Graficamos
pyplot.plot(x, y, "r")
pyplot.show()
```



La predicción es una recta, pero harían falta los datos superpuestos para determinar si el ajuste fue bueno

```
pyplot.scatter(datos["x"], datos["y"])  
pyplot.plot(x, y, "r")  
pyplot.show()
```



Observamos que el ajuste es malísimo, la predicción muestra valores de gasto super altos, mientras que en la realidad son mejores.

Intentemos ajustar β_0 , β_1 a valores más pequeños

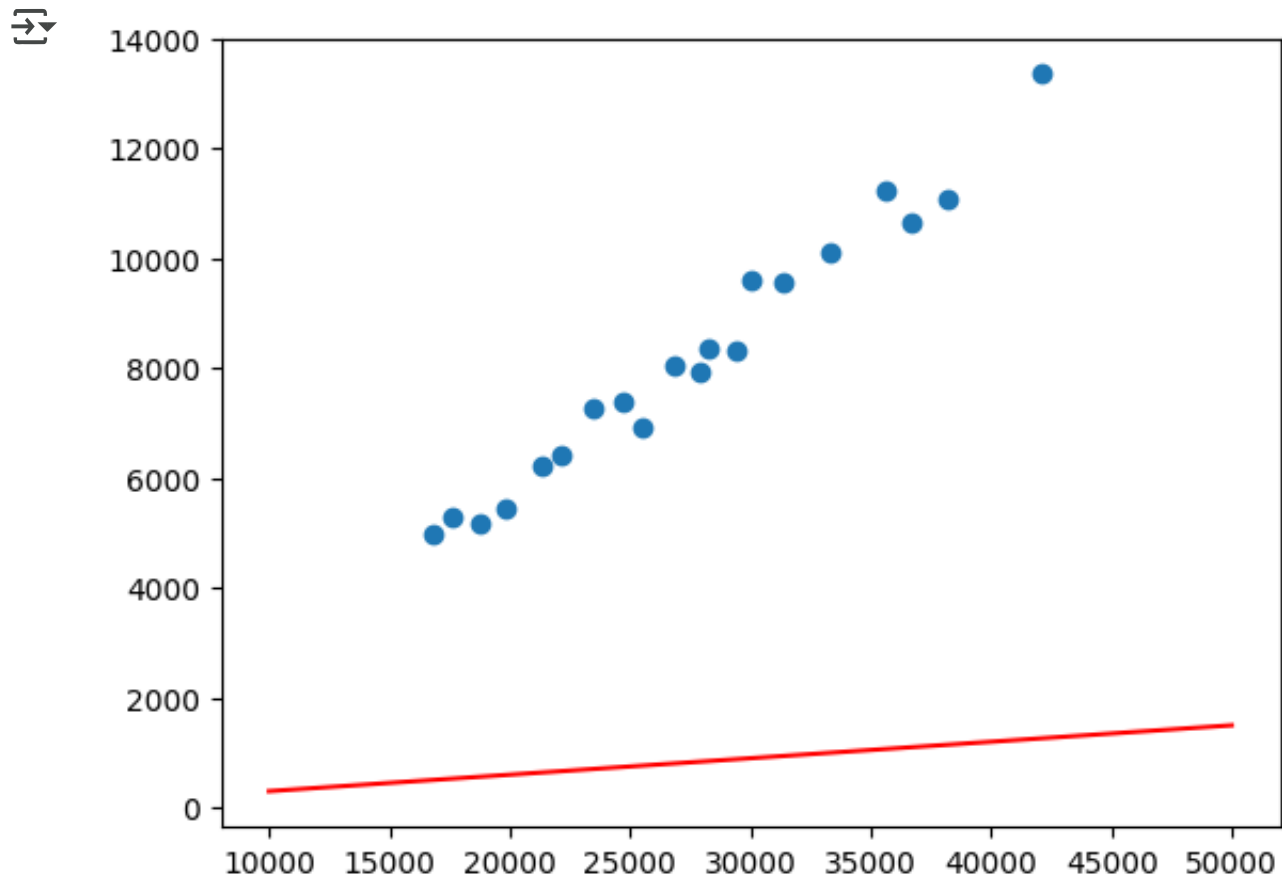
```

beta0 = 0.01
beta1 = 0.03

x = numpy.linspace(10000, 50000, 100)
y = beta0 + beta1 * x

pyplot.scatter(datos["x"], datos["y"])
pyplot.plot(x, y, "r")
pyplot.show()

```



Ahora hemos generado una predicción que no logra predecir más allá de 2, 000

Si observamos un poco mejor, β_1 es la pendiente o incremento que se debería sufrir cada que aumenta x , por ejemplo, si pasamos de 20, 000 a 30, 000 con un aumento en el gasto de 5, 000 a 8, 000 aproximadamente, entonces tendríamos que

$$\beta_1 \approx \frac{8,000-5,000}{30,000-20,000} \approx \frac{3,000}{10,000} = \frac{3}{10} = 0.3$$

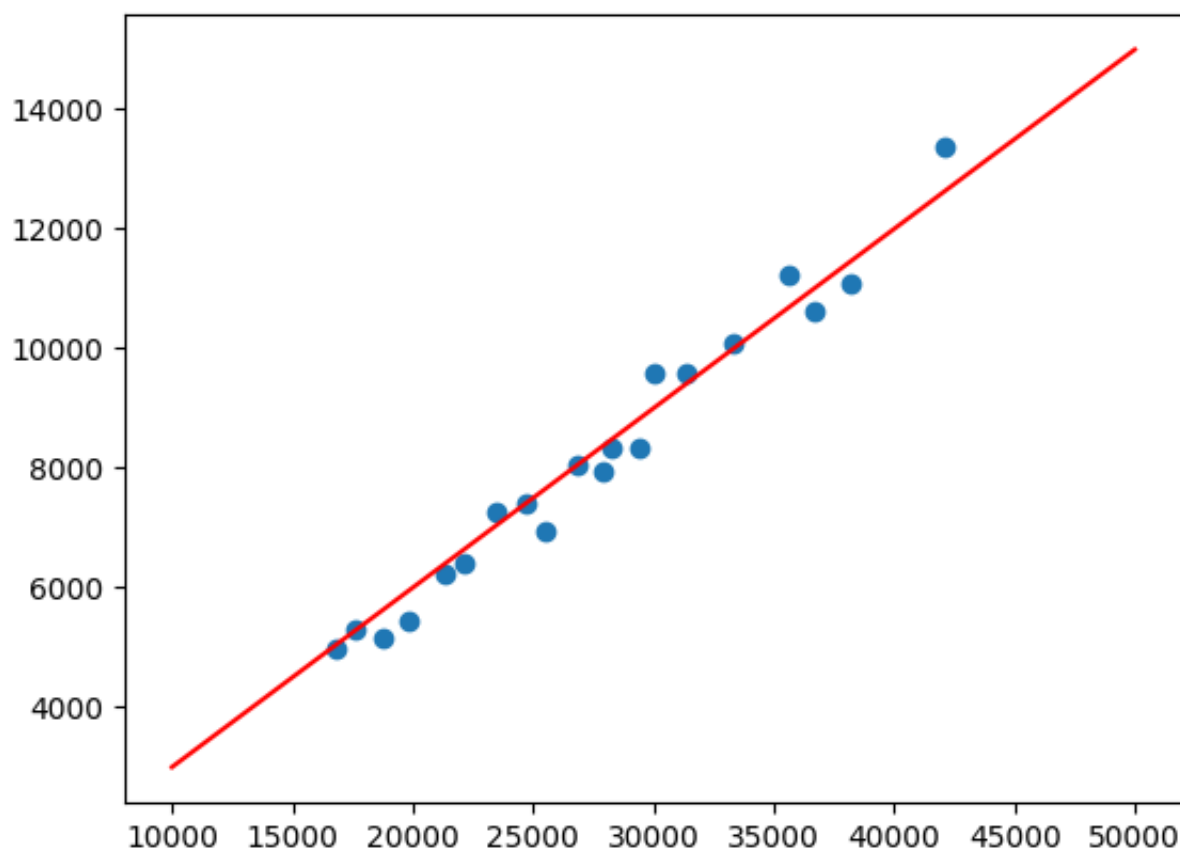
Entonces podríamos pensar que $\beta_1 \approx 0.3$

Proveamos este resultado

```
beta0 = 0
beta1 = 0.3

x = numpy.linspace(10000, 50000, 100)
y = beta0 + beta1 * x

pyplot.scatter(datos["x"], datos["y"])
pyplot.plot(x, y, "r")
pyplot.show()
```



Ahora vemos que la recta se ajusta mejor, pero no es necesariamente la mejor, quizás haya otro β_0, β_1 que se ajuste mejor, además hemos hecho $\beta_0 = 0$, lo cual podría no ser cierto y alguno otro podría resultar mejor.

✓ Regresión Lineal Ingenua

Una estrategia para encontrar β_0, β_1 es buscando de forma ingenua o aleatoria los mejores valores de β_0, β_1 que se ajusten mejor, pero para esto necesitamos medir el error entre la predicción y los datos reales.

Una forma de medir el error entre la predicción y los datos, es mediante el residuo entre la predicción y los datos, si y_i es el i -ésimo dato del eje y , y x_i el i -ésimo dato del eje x , entonces, el valor de predicción para el i -ésimo dato es

$$\hat{y}_i = \beta_0 + \beta_1 \cdot x_i$$

Si calculamos la diferencia entre el dato correcto y_i y el valor de predicción \hat{y}_i para cada i -ésimo dato tendremos el residuo e_i dado por

$$e_i = y_i - \hat{y}_i$$

A la suma de los residuos cuadrados se le conoce como *Suma del Error Cuadrático* o *SSE* dado por

$$SSE = \sum_{i=1}^n e_i = \sum_{i=1}^n y_i - \hat{y}_i$$

Por lo tanto, el *SSE* es una buena medida del error entre el modelo de predicción \hat{y} y los datos reales y .

Por ejemplo, para nuestro modelo de regresión anterior con $\beta_0 = 1$ y $\beta_1 = 3$ tendríamos el siguiente error

```
x = datos["x"]
y = datos["y"]

beta0 = 1
beta1 = 3

yp = beta0 + beta1 * x

sse = sum((y - yp) ** 2)

sse
```

➡ 117446938871.88103

Un error abismalmente grande.

Ahora, para el modelo de regresión anterior con $\beta_0 = 0.01$ y $\beta_1 = 0.03$ tendríamos el siguiente error

```
x = datos["x"]
y = datos["y"]

beta0 = 0.01
beta1 = 0.03

yp = beta0 + beta1 * x

sse = sum((y - yp) ** 2)

sse
```

↔ 1164177851.4417999

Un error abismalmente grande también.

Pero, para el modelo de regresión anterior con $\beta_0 = 0$ y $\beta_1 = 0.3$ tendríamos el siguiente error

```
x = datos["x"]
y = datos["y"]

beta0 = 0.0
beta1 = 0.3

yp = beta0 + beta1 * x

sse = sum((y - yp) ** 2)

sse
```

↔ 3061736.6409999984

Un error aceptablemente menor comparado a los otros, pero quizás demasiado grande aún.

Entonces, para poder encontrar β_0^* y β_1^* óptimos sin algoritmos complejos, podemos hacer una exploración aleatoria de los últimos β_0, β_1 aceptables y calcular su error cuadrático medio (SSE). Entonces, si proponemos nuevos β_0, β_1 que logren tener un SSE menor, estos se irán actualizando hasta que tengamos los mejores β_0^*, β_1^* .

Algoritmo:

1. Proponer un β_0^*, β_1^* inicial con valores pequeños
2. Proponer un β_0, β_1 aleatorios con media β_0^*, β_1^* y varianza flexible (por ejemplo, de 10 o $\sqrt{\sigma_x}$)
3. Calcular el SSE^* para β_0^*, β_1^*
4. Calcular el SSE para β_0, β_1
5. Si el SSE es menor que el SSE^* , entonces actualizamos $\beta_0^* = \beta_0$ y $\beta_1^* = \beta_1$
6. Repetimos del (2) al (5) durante bastantes iteraciones o hasta que ya no cambie β_0^*, β_1^*

Este algoritmo ingenuo, es en realidad un proceso estocástico, que busca actualizar β_0^*, β_1^* hasta que encontremos los mejores que minimicen el SSE^*

Veamos una implementación rápida y lo más natural posible

```

import numpy

x = datos["x"]
y = datos["y"]

# 1. Proponemos beta0* y beta1* (betas óptimos hasta ahora)
beta0_opt = 0
beta1_opt = 0

# Durante 10,000 iteraciones o épocas
for t in range(10000):

    # En cada iteración repetimos los pasos del (2) al (5)

    # 2. Proponemos beta0 y beta1 con distribución normal
    # centrados en beta0* y beta1*
    beta0 = numpy.random.normal(beta0_opt, 100)
    beta1 = numpy.random.normal(beta1_opt, 1)

    # 3. Calcular el SSE* error de los betas óptimos
    yp_opt = beta0_opt + beta1_opt * x
    sse_opt = sum((y - yp_opt) ** 2)

    # 4. Calcular el SSE error de los betas propuestos
    yp = beta0 + beta1 * x
    sse = sum((y - yp) ** 2)

    # 5. Si el SSE es mejor que el SSE*
    # actualizamos los betas óptimos a los propuestos
    if sse < sse_opt:
        beta0_opt = beta0
        beta1_opt = beta1

# Después de todas las iteraciones o épocas
# tenemos los mejores beta0* y beta1* encontrados

beta0_opt, beta1_opt, sse_opt

➡ (-739.0293942817943, 0.3241709757121429, 2528504.415037875)

```

Podemos observar que los mejores parámetros encontrados son $\beta_0 = -739.02$ y $\beta_1 = 0.32$

Nota: Al ser un proceso aleatorio, en cada ejecución podríamos encontrar β_0^* , β_1^* distintos que se ajusten más o menos, con algo de suerte serán similares

Probemos su ajuste:


```
x = datos["x"]
y = datos["y"]

beta0 = -739.0293942817943
beta1 = 0.3241709757121429

yp = beta0 + beta1 * x

sse = sum((y - yp) ** 2)

sse
```

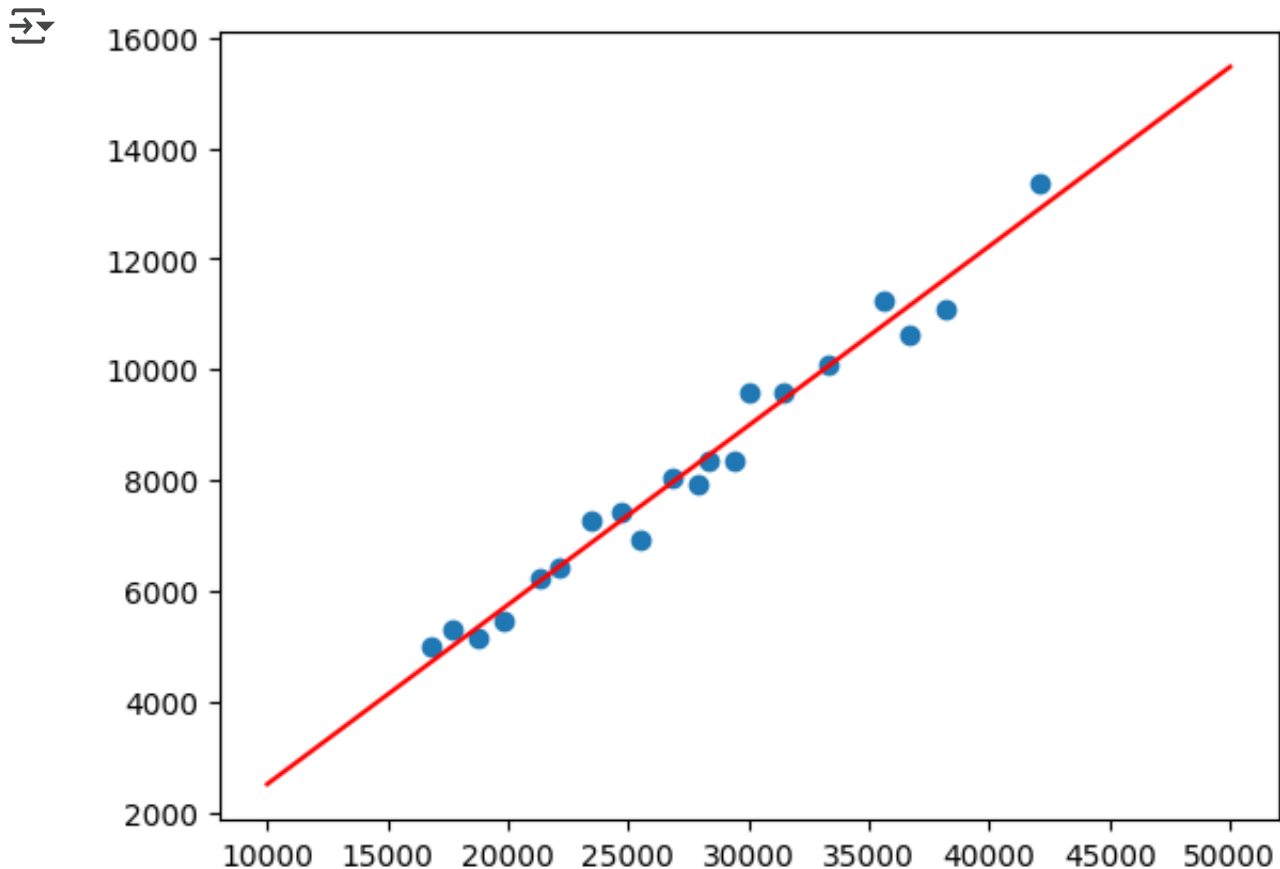
 2528504.415037875

Podemos observar que este error es aún menor que el de $\beta_0 = 0$ y $\beta_1 = 0.3$

La gráfica de ajuste quedaría como

```
x = numpy.linspace(10000, 50000, 100)
y = beta0 + beta1 * x

pyplot.scatter(datos["x"], datos["y"])
pyplot.plot(x, y, "r")
pyplot.show()
```



La cual visualmente se ajusta mejor a los datos

✓ Regresión Lineal por Mínimos Cuadrados Ordinarios

Buscar los parámetros β_0 y β_1 de forma ingenua puede ser lento, ineficiente y muy inexacto, el espacio de exploración es bastante grande como buscar una aguja en un pajar.

Un método más formal para lograr esto es mediante el ajuste por mínimos cuadrados.

Los detalles de este algoritmo los revisaremos en el pizarrón, pero para calcularlos fácilmente podemos usar la librería [Scipy](#)

Esta librería posee herramientas estadísticas, y dentro de estos los métodos de regresión estandarizados que podemos consumir fácilmente, como es la regresión lineal.

El método de regresión lineal por mínimos cuadrados es `LinearRegression`, la cual permite construir un modelo que ajusta automáticamente los

Visita

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html>

```
from scipy.stats import linregress
import pandas

modelo = linregress(x, y)

beta0 = modelo.intercept
beta1 = modelo.slope

pandas.DataFrame({
    "beta0": [beta0],
    "beta1": [beta1]
})
```



	beta0	beta1
0	-649.227946	0.320789



Podemos observar que $\beta_0 = -739.03$ y $\beta_1 = 0.3241$ muy similares a los que encontramos aleatoriamente.

Nota: La optimización también podría diferir de una ejecución a otra, pero en este caso se hará bastante más rápida.

Si analizamos el ajuste tenemos

```
yp = beta0 + beta1 * x

sse = sum((y - yp) ** 2)

sse
```



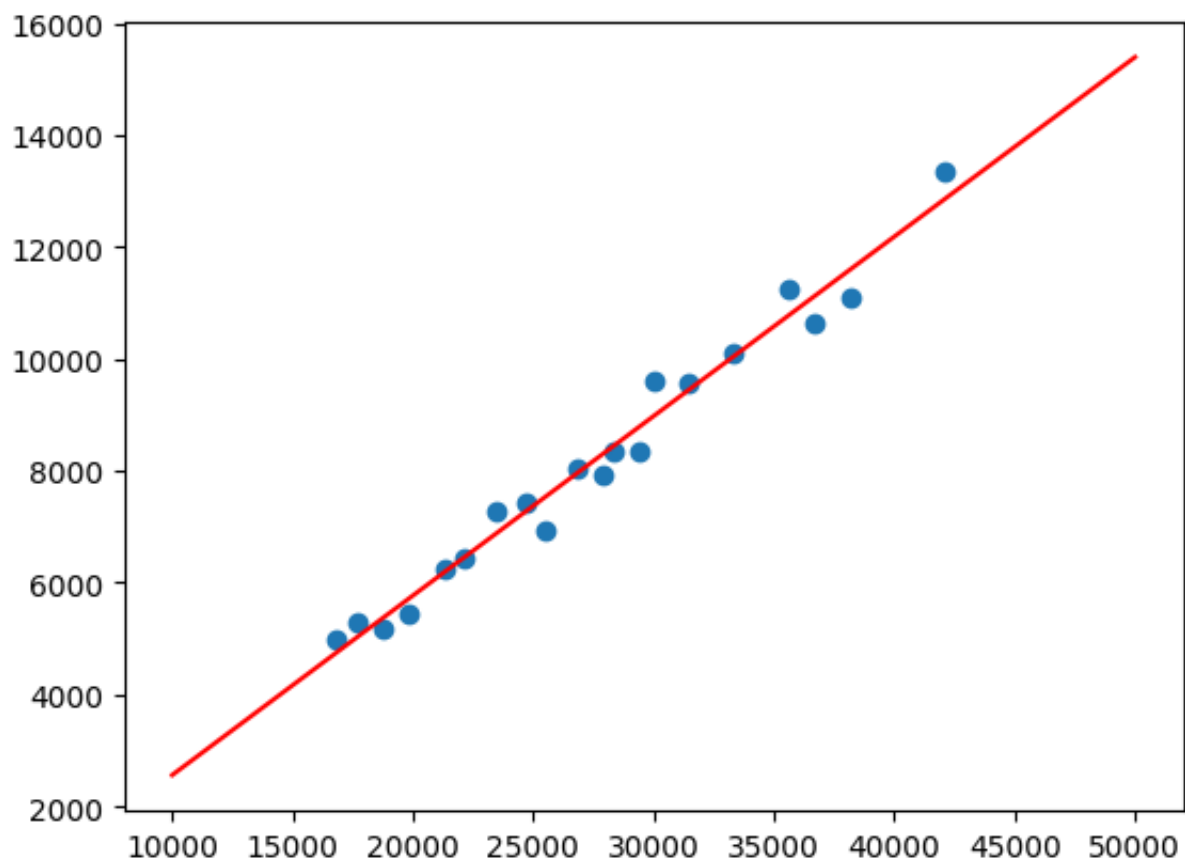
2517087.6438249582

Aunque los resultados son similares, el obtener los parámetros rápidamente es una ventaja, además tenemos una función estandarizada que evita que estemos recordando el proceso estocástico.

Si analizamos el ajuste veremos un ajuste similar

```
x = numpy.linspace(10000, 50000, 100)
y = beta0 + beta1 * x

pyplot.scatter(datos["x"], datos["y"])
pyplot.plot(x, y, "r")
pyplot.show()
```



✓ Modelo de Regresión Lineal Simple

El Modelo de Regresión Lineal Simple es una potente herramienta de análisis e inferencia, este permite predecir la respuesta de una variable y a partir de una variable x .

En la práctica tendremos una forma de lograr predecir datos para una columna o eje de datos, a partir de otra columna.

Algunos ejemplos son:

- **Valuación de Casas:** Se predice el precio de un terreno (variable de respuesta y), mediante la superficie (variable predictora x).
- **Consumo de Combustible:** Se estima la cantidad de combustible consumido por un vehículo (variable de respuesta y) a partir de la distancia recorrida (variable predictora x).
- **Rendimiento Académico:** Se analiza el puntaje final de estudiantes en un examen (variable de respuesta y) con base en las horas de estudio semanales (variable predictora x).

Ahora que tenemos una idea de cómo funciona el Modelo de Regresión Lineal, podemos explotarlo para analizar *datasets*.

(continuación en clase)

✓ Modelo de Regresión Lineal Múltiple

Muchas veces el Modelo de Regresión Lineal Simple no es suficiente, ya que una variable x podría no ser capaz de explicar una variable y .

Por lo tanto, debemos extender nuestro análisis para un modelo que logre predecir el valor de respuesta usando múltiples variables predictoras, en general tendremos:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \dots + \beta_k \cdot x_k$$

(continuación en clase)

✓ Otros Modelos de Regresión

El Machine Learning (Aprendizaje Automático) posee más herramientas de regresión para una predicción mejor en problemas no lineales.

Sin embargo, muchos de estos algoritmos son cajas negras que deben ser estudiados y explorados con cuidado.

Como Analistas o Ingenieros de Datos bastará lograr medir que tan bien se ajustan los modelos y logran las predicciones.

Pero como Científicos de Datos, debemos entender los principios detrás de cada modelo de aprendizaje usado.

[+ Código](#)[+ Texto](#)

(continuación en clase)