untitled0

March 1, 2025

[2]:
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/TaniaGuevara12/Estad-stica-/
 ↪refs/heads/main/data%20(2).csv')
df
```

[2]:

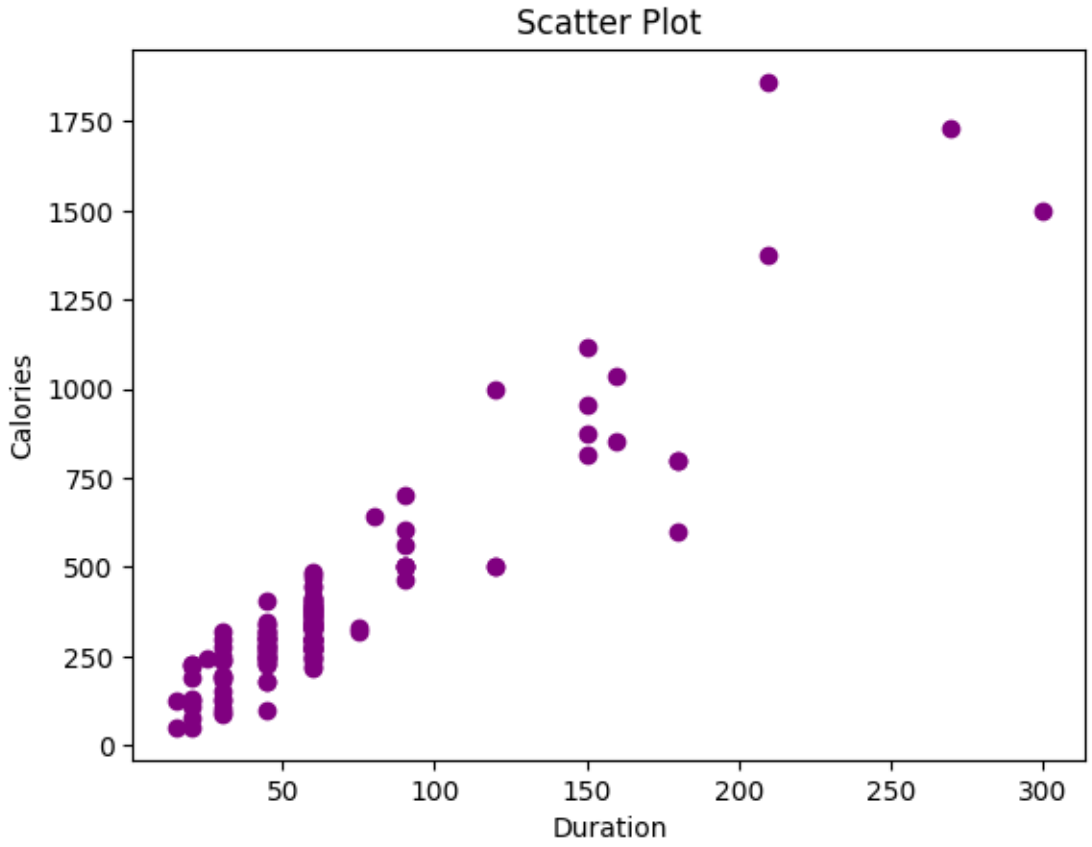
	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
..	…	…	…	…
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

[169 rows x 4 columns]

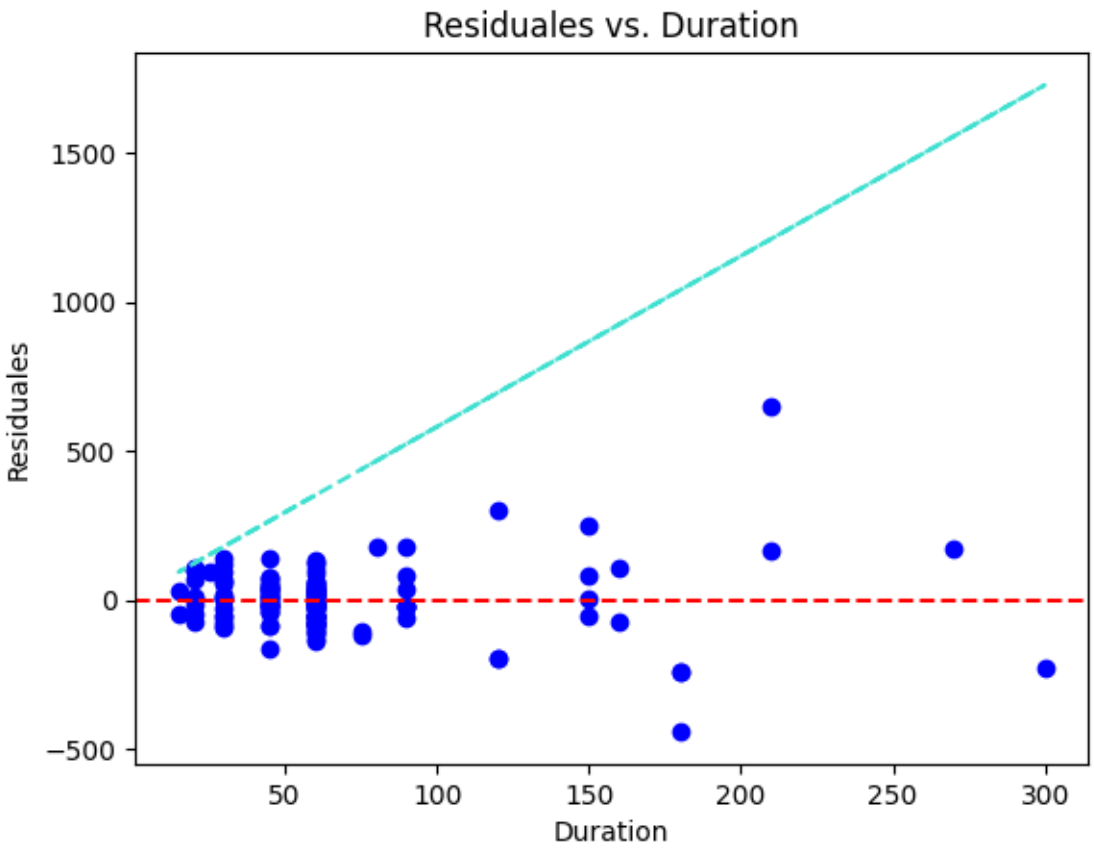
[45]:
```
# a) Establesca una variable dependiente ( Y ) y una variable independiente ( X␣
 ↪).

import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/TaniaGuevara12/Estad-stica-/
 ↪refs/heads/main/data%20(2).csv')
#eliminar registros cpn valores faltantes
df.dropna(inplace=True)

X = df['Duration'] # Variable independiente
Y = df['Calories'] # Variable dependiente

# b) Realiza un gráfico con la dispersión y la recta de regresión ajustada.
import matplotlib.pyplot as plt
plt.scatter(X, Y, color = 'purple')
plt.xlabel('Duration')
plt.ylabel('Calories')
```

```
plt.title('Scatter Plot')
plt.show()

# Recta de regresión lineal.
import statsmodels.api as sm
X_constant = sm.add_constant(X)
model = sm.OLS(Y, X_constant).fit()

b0, b1 = model.params

Fun = lambda x: b0 + b1 * x

Yc= Fun(X)

plt.plot(X, Yc, color = 'turquoise', linestyle = '--')

# C) Calcula el coeficiente de correlación y el coeficiente de determinación e␣
 ↪interpreta los resultados.
from scipy.stats import pearsonr
r,_ = pearsonr(X, Y)
print(f'coeficiente de correlación: {r: 0.4f}\n')
print(f'coeficiente de determinación {r ** 2: 0.4f}\n')

# d) Obtén un intervalo de confianza de 98% para la pendiente e interpreta el␣
 ↪resultado. Respalda tu conclusión usando ANOVA.
nivel_de_confianza = 0.98
intervalo_de_confianza = model.conf_int(alpha = 1 - nivel_de_confianza)
intervalo_de_confianza_b1 = intervalo_de_confianza.iloc[1]
print(f'Intervalo de confianza de {nivel_de_confianza * 100}% para la pendiente:
 ↪')
print(f'{intervalo_de_confianza_b1[0]: 0.4f} < b1 <␣
 ↪{intervalo_de_confianza_b1[1]: 0.4f}')

# Tabla ANOVA
from statsmodels.formula.api import ols
# Y ~ X
modelo_2 = ols('Y ~ X', data = df).fit()
tabla_anova = sm.stats.anova_lm(modelo_2)
print(tabla_anova)

# e) Verifica los supuestos.
residuales = model.resid
plt.scatter(X, residuales, color = 'blue')
plt.xlabel('Duration')
plt.ylabel('Residuales')
plt.title('Residuales vs. Duration')
ax = plt.gca()
```

```
ax.axhline(y = 0, color = 'red', linestyle = '--')
plt.show()

from scipy.stats import shapiro
_, valor_p_shapiro = shapiro(residuales)
print(f'Valor p de Shapiro-Wilk: {valor_p_shapiro: 0.4f}')

from statsmodels.stats.api import het_breuschpagan
_, valor_p_breuschpagan, _, _ = het_breuschpagan(residuales, X_constant)
print(f'Valor p de Breusch-Pagan: {valor_p_breuschpagan: 0.4f}')
```




```
coeficiente de correlación:  0.9227

coeficiente de determinación  0.8514

Intervalo de confianza de 98.0% para la pendiente:
 5.2890 < b1 <  6.1729
             df        sum_sq       mean_sq           F        PR(>F)
X           1.0  9.847530e+06  9.847530e+06  928.219489  5.795220e-69
Residual  162.0  1.718667e+06  1.060905e+04         NaN           NaN
```




```
Valor p de Shapiro-Wilk:  0.0000
Valor p de Breusch-Pagan:  0.0000
```