Trabajo 3

En un estudio a gran escala realizado en EE.UU sobre la eficacia en el control de infecciones hospitalarias se recogió información en 113 hospitales. A su equipo de trabajo le corresponde analizar una muestra aleatoria de n hospitales, que están dentro de un archivo de texto adjunto, donde n es el número de registros en el archivo asignado y X es el número de equipo asignado. La base de datos contiene las siguientes columnas (variables):

- **Y - Riesgo de infección**: Probabilidad promedio estimada de adquirir infección en el hospital (en porcentaje).
- **X1 - Duración de la estadía**: Duración promedio de la estadía de todos los pacientes en el hospital (en días).
- **X2 - Rutina de cultivos**: Razón del número de cultivos realizados en pacientes sin síntomas de infección hospitalaria, por cada 100.
- **X3 - Número de camas**: Número promedio de camas en el hospital durante el periodo del estudio.
- **X4 - Censo promedio diario**: Número promedio de pacientes en el hospital por día durante el periodo del estudio.
- **X5 - Número de enfermeras**: Número promedio de enfermeras, equivalentes a tiempo completo, durante el periodo del estudio.

Se pide:

1. Emplee el análisis de regresión lineal múltiple que explique el riesgo de infección en términos de las variables restantes (actuando como predictoras Xi).
2. Identifique observaciones que puedan considerarse problemáticas (datos atípicos, puntos de balanceo e influyentes) y analice si debe eliminarlas de su conjunto de datos o no, justifique. Repita la construcción del modelo de regresión si eliminó observaciones.
3. Realice la prueba de significancia del modelo, intérprete.
4. Obtener el coeficiente de determinación y el coeficiente de determinación ajustado. Intérprete.
5. Analice si hay problemas de multicolinealidad.
6. Realice una selección de variables por el método que prefiera, tome decisiones, explique.
7. Realice una predicción utilizando el modelo seleccionado, intérprete.

Pasos previos

Inicialmente, se importan las bibliotecas necesarias:

```
# Bibliotecas necesarias
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import itertools

from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import normalize

import scipy.stats as stats
from scipy.stats import bartlett, shapiro
import statsmodels.formula.api as smf
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.stats.stattools import durbin_watson
from statsmodels.stats.outliers_influence import variance_inflation_factor
from IPython.display import display, Markdown
```

Luego, se cargan los datos del archivo para realizar el análisis:

```
raw_data_url =
"https://raw.githubusercontent.com/repos-especializacion-UdeA/estadistica/refs/heads/mai
n/trabajo3/Datos.csv"

# Leer el archivo CSV
df = pd.read_csv(raw_data_url)

# Mostrar las primeras filas del DataFrame
df.head()
```

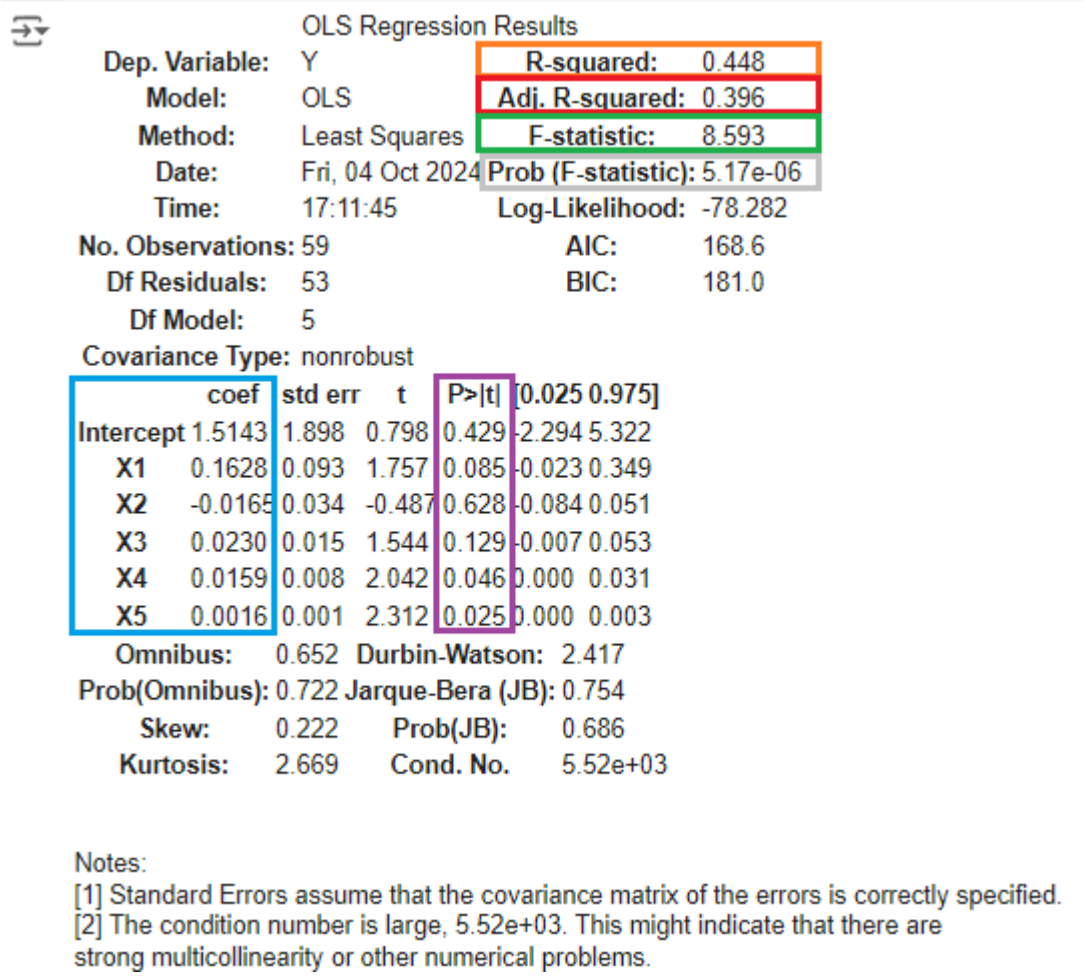
A continuación, se procederá al desarrollo de cada uno de los puntos pedidos:

Punto 1

Emplee el análisis de regresión lineal múltiple que explique el riesgo de infección en términos de las variables restantes (actuando como predictoras **Xi**).

El siguiente código permite obtener el modelo de regresión usando mínimos cuadrados:

```
model = smf.ols(
     formula = 'Y ~ X1 + X2 + X3 + X4 + X5',
     data = df
).fit()
display(model.summary())
```



OLS Regression Results

Dep. Variable:	Y	R-squared:	0.448
Model:	OLS	Adj. R-squared:	0.396
Method:	Least Squares	F-statistic:	8.593
Date:	Fri, 04 Oct 2024	Prob (F-statistic):	5.17e-06
Time:	17:11:45	Log-Likelihood:	-78.282
No. Observations:	59	AIC:	168.6
Df Residuals:	53	BIC:	181.0
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.5143	1.898	0.798	0.429	-2.294	5.322
X1	0.1628	0.093	1.757	0.085	-0.023	0.349
X2	-0.0165	0.034	-0.487	0.628	-0.084	0.051
X3	0.0230	0.015	1.544	0.129	-0.007	0.053
X4	0.0159	0.008	2.042	0.046	0.000	0.031
X5	0.0016	0.001	2.312	0.025	0.000	0.003

Omnibus:	0.652	Durbin-Watson:	2.417
Prob(Omnibus):	0.722	Jarque-Bera (JB):	0.754
Skew:	0.222	Prob(JB):	0.686
Kurtosis:	2.669	Cond. No.	5.52e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.52e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Teniendo en cuenta la información resaltada en el recuadro azul, el modelo resultante está definido por:

$$Y=1.5143+0.1628X1-0.0165X2+0.0230X3+0.0159X4+0.0016X5$$

La siguiente tabla resume los resultados obtenidos de la figura anterior:

Métrica	**Resultado**
Error estándar de los residuos (Residual Standard Error, RSE)	`Adj. R-squared:  0.396`
Coeficiente de determinación	`R-squared:  0.448`
Estadístico F	`F-statistic:  8.593`
Valor p (VP)	`Prob (F-statistic):  5.17e-06`

Respecto a la tabla anterior el valor de R^2 nos indica que cerca del 44.8% de la variabilidad de los datos es explicada por este modelo.

Como el valor del estadístico F = 8.593 es alto y el valor VP = 5.17e-16 es mucho menor que el el nivel de significancia (alpha = 0.05) podemos decir que el modelo lineal es significativo en su conjunto, lo cual implica que que al menos uno de los predictores (X1, X2, X3, X4 o X5) tiene un impacto significativo en la variable dependiente Y.

Para determinar cuales variables predictoras son significativas o no, se emplea la siguiente prueba de hipótesis usando un alpha de 0.05:

- **El predictor es estadísticamente significativo**: Si VP < alpha
- **El predictor es estadísticamente no significativo**: Si VP > alpha

El siguiente fragmento de código permite obtener las variables significativas al aplicar la anterior prueba de hipótesis teniendo en cuenta lo anterior:

```
# Crear un DataFrame con los valores p

p_values_df = pd.DataFrame(model.pvalues, columns=['p_value'])
p_values_df

significativo = lambda p_value, alpha=0.05: True if p_value < alpha else False
p_values_df['significancia'] = p_values_df['p_value'].apply(significativo)
p_values_df
```

La salida del código anterior se muestra en la siguiente figura:

	p_value	significancia
Intercept	0.428630	False
X1	0.084665	False
X2	0.628346	False
X3	0.128542	False
X4	0.046120	True
X5	0.024718	True

Según los resultados anteriores podemos concluir que:

- Los parámetros B0, B1, B2 y B3 no son significativos para el modelo.
- Los parámetros (relacionados con los predictores X4 y X4) B4 y B5 son estadísticamente significativos con un nivel de confianza del 95%.

Finalmente, contextualizando esto al problema tenemos que en lo que respecta al modelo a mayor número de pacientes (X4) y mayor número de enfermeras (X5) el riesgo de infección (Y) aumenta.

Punto 2

Identifique observaciones que puedan considerarse problemáticas (datos atípicos, puntos de balanceo e influyentes) y analice si debe eliminarlas de su conjunto de datos o no, justifique. Repita la construcción del modelo de regresión si eliminó observaciones.

Para la identificación de observaciones que pueden ser problemáticas se procedio a realizar el análisis de los datos con el fin de identificar:

1. Observaciones atípicas (outliers)
2. Puntos de balanceo.
3. Observaciones influenciables.

El siguiente código resume en una tabla los puntos fundamentales para el caso:

```
# Calcular la distancia de Cook
influencia = model.get_influence()

# Calcular la distancia de Cook
cooks_d = influencia.cooks_distance[0]

# Calcular los valores de apalancamiento (hii)
hii_values = influencia.hat_matrix_diag

# Calcular los residuales estudentizados
rstudent_values = influencia.resid_studentized_external

# Calcular los valores DFFITS
dffits_values = influencia.dffits[0]

# Calcular los valores DFbeta
dfbeta_values = influencia.dfbetas

# Crear un DataFrame con todos los valores de influencia
influence_df = pd.DataFrame({
    'Leverage (hii)': hii_values,
    'Cooks Distance': cooks_d,
    'R-Student': rstudent_values,
    'DFFITS': dffits_values
    # Puedes agregar más métricas si lo deseas
})

# Mostrar el DataFrame
display(influence_df)
```

El resultado se muestra en la siguiente tabla (resaltando en las filas resaltadas en amarillo los datos sospechosos):

i	Leverage (hii)	Cooks Distance	R-Student	DFFITS
0	0.18036	0.001202	0.179408	0.084159
1	0.057051	6.50E-05	-0.079473	-0.019548
2	0.051793	0.002259	-0.494597	-0.115594
3	0.050941	0.001412	-0.394105	-0.091306
4	0.077732	0.014745	1.025023	0.297581
5	0.064841	0.026946	-1.546928	-0.407336

6	0.109914	0.00262	-0.353848	-0.124345
7	0.126799	0.01905	0.885394	0.337393
8	0.027164	0.000139	-0.171543	-0.028665
9	0.040716	0.001995	0.527434	0.108662
10	0.112809	0.006739	-0.560241	-0.199774
11	0.090248	0.000556	-0.181677	-0.057221
12	0.108304	0.064518	-1.824034	-0.635691
13	0.128192	0.03388	1.180124	0.452531
14	0.028603	0.001024	0.453353	0.077794
15	0.118033	0.026318	-1.088122	-0.398063
16	0.195564	0.000575	-0.117981	-0.058172
17	0.131804	0.025049	0.994893	0.387642
18	0.171241	0.001217	0.186269	0.08467
19	0.088783	0.03931	-1.577566	-0.492427
20	0.046241	0.003924	0.693449	0.152689
21	0.128524	0.0002	-0.089368	-0.03432
22	0.088202	0.000408	-0.157695	-0.049046
23	0.056739	0.001052	-0.321162	-0.078768
24	0.249065	0.143863	-1.638671	-0.943729
25	0.132593	0.005852	0.47577	0.186014
26	0.081801	0.042442	1.721732	0.513898
27	0.097987	0.003715	-0.449582	-0.148179
28	0.408237	0.218789	-1.391585	-1.155824
29	0.08644	0.045918	-1.738646	-0.534811
30	0.037468	0.019907	1.787602	0.352692
31	0.194983	0.059041	-1.214777	-0.59785

32	0.064932	0.000606	0.226813	0.059769
33	0.051696	0.009951	-1.047511	-0.244575
34	0.067648	0.010551	0.932947	0.251301
35	0.058995	0.012055	1.075715	0.269346
36	0.206046	0.11062	-1.623723	-0.827174
37	0.044641	0.008602	1.052032	0.227412
38	0.040542	0.019769	1.705317	0.350548
39	0.130139	0.183639	2.896877	1.120493
40	0.043865	0.019244	-1.610114	-0.34487
41	0.07776	0.045057	1.829849	0.531338
42	0.081538	0.004147	-0.525768	-0.156655
43	0.096738	0.018962	-1.031309	-0.337504
44	0.066537	0.003862	-0.566507	-0.151248
45	0.061628	0.000207	-0.136085	-0.034875
46	0.055455	0.000153	0.123803	0.029998
47	0.435206	0.229908	-1.348265	-1.183525
48	0.027776	0.001473	0.552451	0.093379
49	0.092568	0.002966	0.41439	0.132352
50	0.072273	0.000938	0.266396	0.074354
51	0.04585	0.005356	-0.815181	-0.178695
52	0.026138	0.000446	-0.312969	-0.051273
53	0.062983	0.01266	1.06437	0.275951
54	0.058823	0.00098	0.30408	0.076019
55	0.140354	0.010813	0.626748	0.253247
56	0.190383	3.90E-05	-0.03111	-0.015086
57	0.087178	8.50E-05	-0.072579	-0.02243

58	0.043135	0.002323	0.552438	0.117294

A continuación de describen los diferentes analisi realizados para determinar cuales fueron los datos sospechosos:

Outliers

Para la determinación de los outliers se utilizaron los residuales estudentizados (Columna R-student) siguiendo el siguiente criterio:

- Si el residual estudentizado es mayor a 2 o menor a -2 es considerado sospechoso.
- Si el residual estudentizado mayor a 3 o menor a -3 se considera un outlier significativo.

El siguiente fragmento de código aplica este criterio a la tabla anterior:

```
mask_outliers = influence_df['R-Student'].apply(lambda x: True if abs(x) > 2 else False)
outliers = influence_df[mask_outliers]
outliers
```

El resultado se muestra a continuación:

i	Leverage (hii)	Cooks Distance	R-Student	DFFITS
39	0.130139	0.183639	2.896877	1.120493

Según lo anterior, el dato correspondiente a la muestra 40 (índice 39) es outlier.

Puntos de de balanceo

Para obtener los puntos de balanceo se empleó el criterio de leverage el cual establece que para la observación **i** es un **punto de balanceo** si su **hii > 2p/n**. Para nuestro caso:

- Número de parámetros del modelo: p = 5 (X1, X2, X3, X4 y X5).
- Número de observaciones: n = 59.

De modo que cada valor de hii se está comparando con el **umbral[1] = 2(p+1)/n = 2*6/59 = 0.2033** para la determinación del punto de balanceo tal y como lo muestra el siguiente fragmento de código:

```
leverage_threshold = 2 * (len(df.columns)) / len(df)
high_leverage_points = df[hii_values > leverage_threshold]
high_leverage_points
```

[1] A diferencia del umbral 2n/p, el umbral 2(n+1)/p es ligeramente más flexible y puede ser preferido en análisis con conjuntos de datos pequeños.

La siguiente tabla muestra los puntos de balanceo teniendo en cuenta el criterio anterior para los valores de hii (columna resaltada).

i	Leverage (hii)	Cooks Distance	R-Student	DFFITS
24	0.249065	0.143863	-1.638671	-0.943729
28	0.408237	0.218789	-1.391585	-1.155824
36	0.206046	0.11062	-1.623723	-0.827174
47	0.435206	0.229908	-1.348265	-1.183525

Tratar los puntos de balanceo es importante ya que estos pueden distorsionar los resultados del modelo de regresión, sin embargo, la decisión de eliminarlos o no depende del contexto del problema.

Puntos de influencia

Un punto de influencia es una observación en un conjunto de datos que tiene un impacto desproporcionado en los resultados del modelo de regresión. se dice que la observación i será influencial si la distancia cook es superior a 1. A continuación se muestra el fragmento del código python que hace esto:

```
mask_influence_points = influence_df['Cooks Distance'] > 1
influence_points = influence_df[mask_influence_points]
influence_points
```

El resultado muestra que para este caso, no hay:

i	Leverage (hii)	Cooks Distance	R-Student	DFFITS

La siguiente tabla resume las características de los puntos encontrados:

i	Punto outlier	Punto de balanceo	Influenciable
24		X	
28		X	
36		X	
39	X		
47		X	

Para ver si eliminamos los puntos considerados sospechosos obtuvimos la media de para las variables predictoras más significativas (X4 y X5):

```
df[['X4','X5']].mean()
```

El resultado fue:

- media(X4) = 81.449153
- media(X5) = 281.525424

A continuación, la lista de puntos sospechosos es la siguiente:

i	X1	X2	X3	X4	X5
24	10.79	44.2	2.6	56.6	461
28	17.94	56.2	26.4	91.8	835
36	11.07	53.2	28.5	122.0	768
39	11.41	61.1	16.6	97.9	535
47	11.18	45.7	60.5	85.8	640

Aunque no conocemos mucho del contexto del problema, apelando a la comparación de las medias con los puntos considerados sospechosos (sobre todo para la última columna), nuestra corazonada se inclina por eliminar estos puntos ya que se encuentran los valores de la característica X5 se encuentran bastante alejados de la media de X5.

A continuación, se muestra el código para eliminar los datos sospechosos.

```
non_influential_data = df.drop([24, 28, 36, 39, 47])
non_influential_data
```

Ahora, a partir de los datos anteriores, se procederá a obtener el modelo:

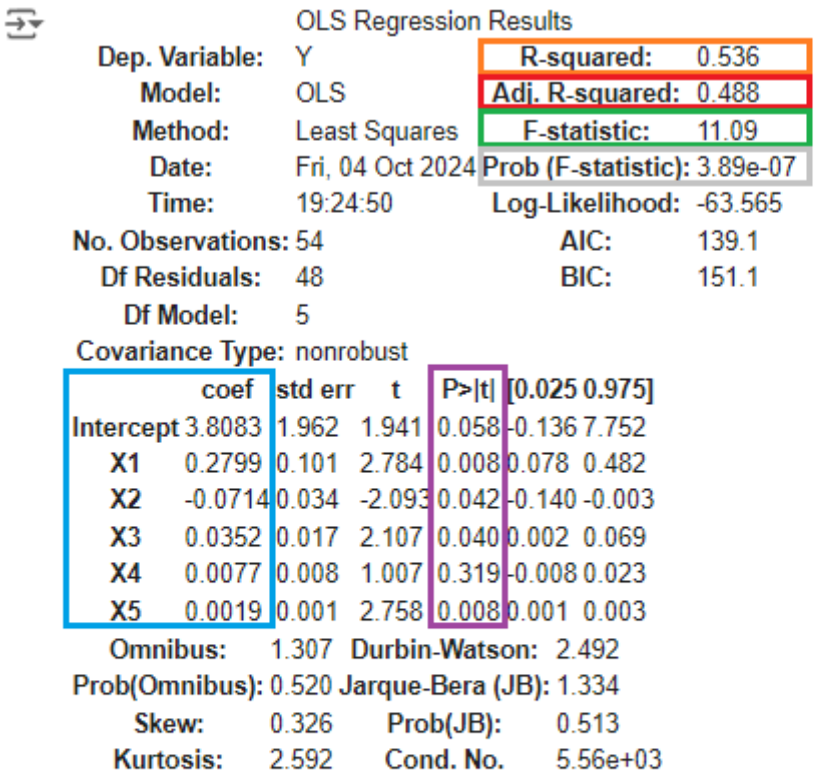
```
new_model = smf.ols('Y ~ X1 + X2 + X3 + X4 + X5', data = non_influential_data).fit()
```

Punto 3

Realice la prueba de significancia del modelo, intérprete.

```
new_model.summary()
```

El resultado se muestra en la figura anterior



OLS Regression Results

Dep. Variable:	Y	R-squared:	0.536
Model:	OLS	Adj. R-squared:	0.488
Method:	Least Squares	F-statistic:	11.09
Date:	Fri, 04 Oct 2024	Prob (F-statistic):	3.89e-07
Time:	19:24:50	Log-Likelihood:	-63.565
No. Observations:	54	AIC:	139.1
Df Residuals:	48	BIC:	151.1
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.8083	1.962	1.941	0.058	-0.136	7.752
X1	0.2799	0.101	2.784	0.008	0.078	0.482
X2	-0.0714	0.034	-2.093	0.042	-0.140	-0.003
X3	0.0352	0.017	2.107	0.040	0.002	0.069
X4	0.0077	0.008	1.007	0.319	-0.008	0.023
X5	0.0019	0.001	2.758	0.008	0.001	0.003

Omnibus:	1.307	Durbin-Watson:	2.492
Prob(Omnibus):	0.520	Jarque-Bera (JB):	1.334
Skew:	0.326	Prob(JB):	0.513
Kurtosis:	2.592	Cond. No.	5.56e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.56e+03. This might indicate that there are strong multicollinearity or other numerical problems.

La siguiente tabla resume los resultados obtenidos de la figura anterior:

Métrica	**Resultado**
Error estándar de los residuos (Residual Standard Error, RSE)	`Adj. R-squared:  0.536`
Coeficiente de determinación	`R-squared:  0.448`
Estadístico F	`F-statistic:  11.09`
Valor p (VP)	`Prob (F-statistic):  3.89e-07`

Como el VP = 3.89e-07 es mucho menor que alpha = 0.05 y el valor del estadístico F = 11.09 es alto, podemos decir que el modelo lineal es significativo en su conjunto. Ademas, teniendo en cuenta un nivel de significancia de alpha = 5%, llegamos a que las variables predictoras que pueden considerarse significativas (que cumplen que VP < alpha) son: **X1**, **X2**, **X3** y **X5**.

Punto 4

Obtener el coeficiente de determinación y el coeficiente de determinación ajustado. Interprete.

De la Tabla anterior se puede ver que:

- Coeficiente de determinación: 53.6%
- Coeficiente de determinación ajustado: 44.8 %

Lo anterior implica que la variabilidad explicada por el modelo es del 53.6%. La siguiente tabla compara los valores de estos coeficientes respecto al modelo inicial (en el que no se eliminaron los puntos):

Modelo	**Coeficiente de determinación (%)**	**Coeficiente de determinación ajustado (%)**
Con todos los puntos	44.8	39.6
Eliminando los sospechosos	53.6	44.8

Al comparar los resultados de este nuevo modelo respecto al anterior podemos ver que un mejor ajuste pues explica un poco mejor la variabilidad de los datos.

Punto 5

En el siguiente código python se calculan los VIF:

```
### Factores de Inflación de Varianza

X_non_influential_data = non_influential_data[['X1','X2','X3','X4','X5']]
X_with_const = sm.add_constant(X_non_influential_data)

vif = pd.DataFrame()
vif["Variable"] = X_non_influential_data.columns
vif["VIF"] = [variance_inflation_factor(X_with_const.values, i) for i in range(1,
X_with_const.shape[1])]

vif
```

La siguiente tabla muestra los resultados:

Variable	**VIF**
X1	1.498908
X2	1.220344
X3	1.420350

X4	1.553573
X5	1.250247

Para analizar si existen problemas de multicolinealidad se empleó el análisis de Factores de Inflación de Varianza donde se sigue el siguiente criterio:

- Si VIF_j < 5 no hay multicolinealidad.
- Si 5 < VIF_j < 10 hay multicolinealidad moderada.
- Si VIF_j < 10 hay multicolinealidad grave.

El análisis VIF muestra que no hay multicolinealidad entre las variables independientes y por lo tanto las variables que se incluyen en el modelo explican bien la variabilidad de la variable Y.

Paso 6

Selección de variables

La siguiente función crea la tabla para todas las regresiones posibles:

```
def todas_regresiones_posibles_completo(data, respuesta):
    variables_predictoras = list(data.columns)
    variables_predictoras.remove(respuesta)

    n = len(data)
    resultados = []

    # Probar todas las combinaciones de variables predictoras
    for k in range(1, len(variables_predictoras) + 1):
        for combo in itertools.combinations(variables_predictoras, k):
            # Definir el conjunto de predictores actual
            X = data[list(combo)]
            X = sm.add_constant(X)  # Agregar la constante (intercepto)
            y = data[respuesta]

            # Ajustar el modelo de regresión
            modelo = sm.OLS(y, X).fit()

            # Obtener los valores necesarios
            R2 = modelo.rsquared
            R2_adj = modelo.rsquared_adj
            SSE = np.sum(modelo.resid ** 2)  # Suma de cuadrados del error
            MSE = SSE / modelo.df_resid  # Error cuadrático medio
            Cp = SSE / MSE - (n - 2 * (k + 1))  # Criterio de Cp de Mallows

            # Guardar los resultados del modelo
            resultados.append({
                'Predictoras': combo,
                'R2': R2,
                'R2_adj': R2_adj,
                'SSE': SSE,
                'MSE': MSE,
```

```
                'Cp': Cp,
                'AIC': modelo.aic,
                'BIC': modelo.bic
            })

    # Convertir los resultados a un DataFrame
    resultados_df = pd.DataFrame(resultados)

    return resultados_df
```

Aplicando la función anterior tenemos los resultados para los distintos modelos que se pueden obtener a partir de los predictores a partir de los datos filtrados organizando los datos de acuerdo al valor de R2 de mayor a menor:

```
resultados_posibles = todas_regresiones_posibles_completo(non_influential_data, 'Y')
resultados_posibles = resultados_posibles.sort_values(by='R2_adj', ascending=False)
resultados_posibles
```

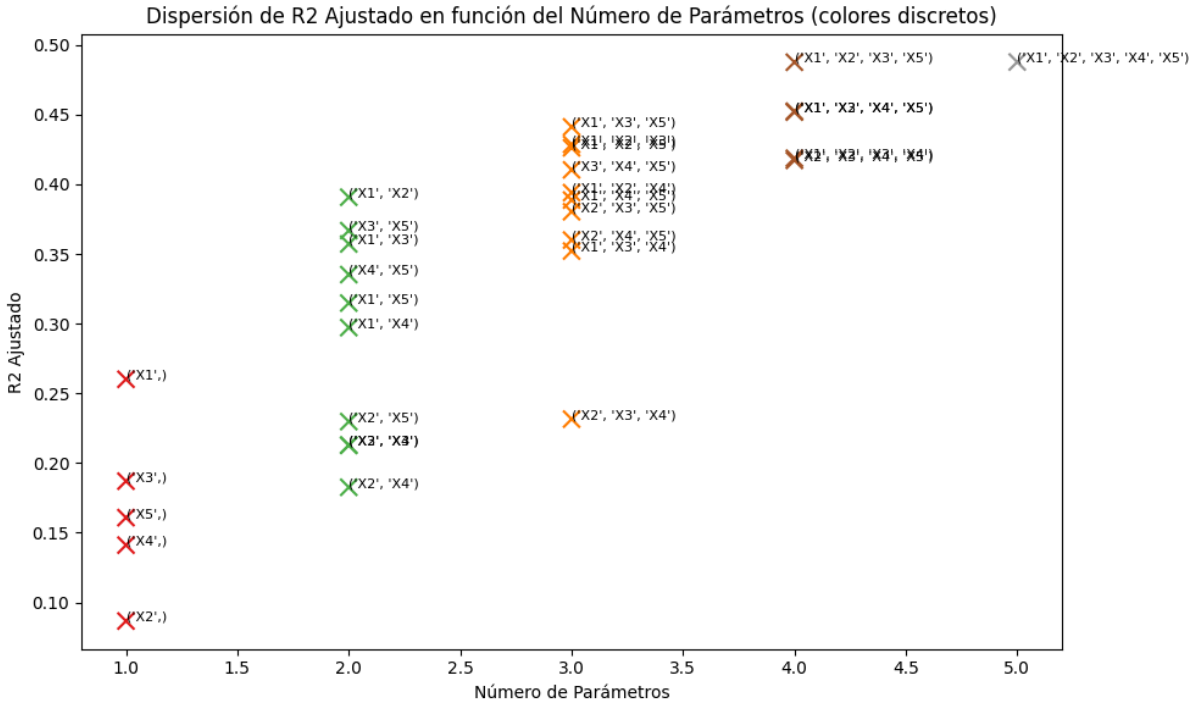
La tabla de todas las regresiones posibles resultante se muestra a continuación:

Predictoras	R2	R2_adj	SSE	MSE	Cp	AIC	BIC
(X1, X2, X3, X4, X5)	0.536129	0.487799	33.293628	0.693617	6	139.129988	151.063892
(X1, X2, X3, X5)	0.526311	0.487642	33.997625	0.693829	5	138.25992	148.20484
(X1, X3, X4, X5)	0.493804	0.452482	36.330708	0.741443	5	141.844043	151.788964
(X1, X2, X4, X5)	0.493226	0.451856	36.372236	0.742291	5	141.905733	151.850653
(X1, X3, X5)	0.472859	0.44123	37.834018	0.75668	4	142.03349	149.989426
(X1, X2, X3)	0.461059	0.428722	38.680913	0.773618	4	143.228922	151.184858
(X1, X2, X5)	0.458695	0.426217	38.850536	0.777011	4	143.465204	151.42114
(X1, X2, X3, X4)	0.462606	0.418737	38.569884	0.78714	5	145.073699	155.018619
(X2, X3, X4, X5)	0.461208	0.417225	38.670209	0.789188	5	145.213977	155.158897
(X3, X4, X5)	0.443768	0.410394	39.921898	0.798438	4	144.934173	152.890109

(X1, X2, X4)	0.428369	0.394071	41.027151	0.820543	4	146.408862	154.364798
(X1, X2)	0.413761	0.390771	42.07556	0.825011	3	145.771444	151.738396
(X1, X4, X5)	0.423359	0.388761	41.386696	0.827734	4	146.880035	154.835971
(X2, X3, X5)	0.415442	0.380368	41.954939	0.839099	4	147.616416	155.572352
(X3, X5)	0.390696	0.366802	43.730987	0.85747	3	147.855298	153.822 25
(X2, X4, X5)	0.396272	0.360048	43.330781	0.866616	4	149.358 84	157.314 776
(X1, X3)	0.381503	0.357249	44.390756	0.870407	3	148.663 911	154.630 863
(X1, X3, X4)	0.388833	0.352163	43.864693	0.877294	4	150.020 149	157.976 085
(X4, X5)	0.360362	0.335278	45.908135	0.90016	3	150.478 909	156.445 861
(X1, X5)	0.340679	0.314824	47.320789	0.927859	3	152.115 508	158.082 46
(X1, X4)	0.323732	0.297212	48.537108	0.951708	3	153.485 968	159.452 92
(X1,)	0.274102	0.260142	52.099192	1.001908	2	155.310 293	159.288 261
(X2, X3, X4)	0.275029	0.231531	52.032648	1.040653	4	159.241 277	167.197 213
(X2, X5)	0.258841	0.229776	53.194516	1.04303	3	158.433 81	164.400 762
(X3, X4)	0.242639	0.212938	54.357363	1.065831	3	159.601 547	165.568 499
(X2, X3)	0.242443	0.212735	54.371396	1.066106	3	159.615 486	165.582 438
(X3,)	0.202374	0.187035	57.247221	1.100908	2	160.398 695	164.376 663
(X2, X4)	0.213468	0.182623	56.451035	1.106883	3	161.642 398	167.609 351

(X5,)	0.176686	0.160853	59.090891	1.136363	2	162.110369	166.088337
(X4,)	0.157333	0.141128	60.479936	1.163076	2	163.365054	167.343022
(X2,)	0.103893	0.08666	64.315413	1.236835	2	166.685386	170.663354

A continuación se muestra la gráfica que compara el número de parámetros contra R^2 para cada combinación posible de variables.



A partir de la información anterior, seleccionamos como mejor modelo aquel con el menor número de variables predictoras con el mayor R^2 de manera este no cambie significativamente cuando se aumenta la cantidad de variables predictoras. Teniendo en cuenta lo anterior, el mejor modelo es el que tiene como predictores X1, X2, X3 y X5.

A continuación se muestran las instrucciones de python para generar modelo teniendo en cuenta los hallazgos anteriores:

```
X_final = X_non_influential_data[['X1', 'X2', 'X3', 'X5']]
X_final = sm.add_constant(X_final)
Y_final = non_influential_data[['Y']]

# Ajustar el modelo de regresión
model_final = sm.OLS(Y_final, X_final).fit()

# Mostrar el resumen del modelo
model_final.summary()
```

OLS Regression Results

Dep. Variable:	Y	R-squared:	0.526
Model:	OLS	Adj. R-squared:	0.488
Method:	Least Squares	F-statistic:	13.61
Date:	Sat, 05 Oct 2024	Prob (F-statistic):	1.56e-07
Time:	03:27:15	Log-Likelihood:	-64.130
No. Observations:	54	AIC:	138.3
Df Residuals:	49	BIC:	148.2
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.4038	1.871	2.354	0.023	0.645	8.163
X1	0.3170	0.094	3.387	0.001	0.129	0.505
X2	-0.0785	0.033	-2.351	0.023	-0.146	-0.011
X3	0.0413	0.016	2.645	0.011	0.010	0.073
X5	0.0017	0.001	2.598	0.012	0.000	0.003

Omnibus:	0.877	Durbin-Watson:	2.399
Prob(Omnibus):	0.645	Jarque-Bera (JB):	0.933
Skew:	0.276	Prob(JB):	0.627
Kurtosis:	2.670	Cond. No.	5.19e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.19e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Para empezar tenemos que el modelo en su conjunto es estadísticamente significativo (**VP << alpha → 1.56e-07 << 0.05**). En lo que respecta al valor de **R^2 = 0.526** el modelo explica el **52.6%** de la variabilidad en la variable dependiente (**Y**) de modo que el ajuste fue bueno (respecto al modelo original), sin embargo, aún el modelo se queda corto. Así mismo vemos que, todas las variables predictoras empleadas: **X1**, **X2**, **X3** y **X5** son significativas pues el valor p asociada a cada una de estas es menor que 0.05 (cuadro morado). De este modo el modelo lineal multivariado quedó como:

```
Y= 4.4038 + 0.3170X1 - 0.0785X2 + 0.0413X3 + 0.0017X5
```

Respecto a la validación de supuestos del modelo tenemos que:

- Los errores son independientes ya que el valor estadístico de Durbin-Watson = 2.399 es cercano a 2 lo cual implica que estos no están correlacionados.
- Los errores siguen una distribución normal pues el valor arrojado por el valor p para la prueba **Prob(obnibus) > alpha → 0.645 > 0.05** (de modo que no hay suficiente evidencia para rechazar la hipótesis nula: Los residuos del modelo siguen una distribución normal).

- Como el número de condición **k** es muy alto **Cond. No. = 5.19e03** > **30** existe una colinealidad muy alta, por lo que este supuesto no se cumple.

Punto 7

Realice una predicción utilizando el modelo seleccionado, intérprete.

Para realizar la predicción del modelo se emplearon los datos originales y se generó una columna a la salida en la cual mostraron las diez primeras predicciones comparadas con los datos que se esperaban. El código que hace esto se muestra a continuación:

```
# Datos de entrada del modelo (los originales)
X_original = pd.DataFrame({
    'X1': df['X1'],
    'X2': df['X2'],
    'X3': df['X3'],
    'X5': df['X5']
})

# Agregar la constante
X_original = sm.add_constant(X_original)

# Realizar la predicción
predictions_final_model = model_final.predict(X_original)

# Mostrar las predicciones
comp = pd.DataFrame()
comp['Y'] = df['Y']
comp['Y_pred'] = predictions_final_model
# Se muestran los 10 primeros elementos de la tabla
comp.head(10)
```

Los primeros 10 datos se muestran a continuación:

	Y	Y_pred
0	4.8	4.151870
1	5.0	5.333661
2	4.4	5.004486
3	3.7	3.859206
4	5.5	5.397823
5	1.7	2.968544
6	4.5	5.331876
7	5.7	5.245321
8	4.0	4.448194
9	4.2	3.640535

Vemos que aunque las predicciones generadas por el modelo están relativamente cercanas a los valores reales hay algunas discrepancias, lo que podría indicar que hay cierta variabilidad en los datos que el modelo no está capturando completamente. Lo cual se evidencio en el valor de R^2 aún muy lejano de un 90% si quiera y en el hecho de que al parecer seguía habiendo colinealidad entre las variables.

Notebook: En notebook relacionado se encuentra es **trabajo3.ipynb** (link)