Tarea

September 26, 2024

1 ACTIVIDAD. Matriz HAT

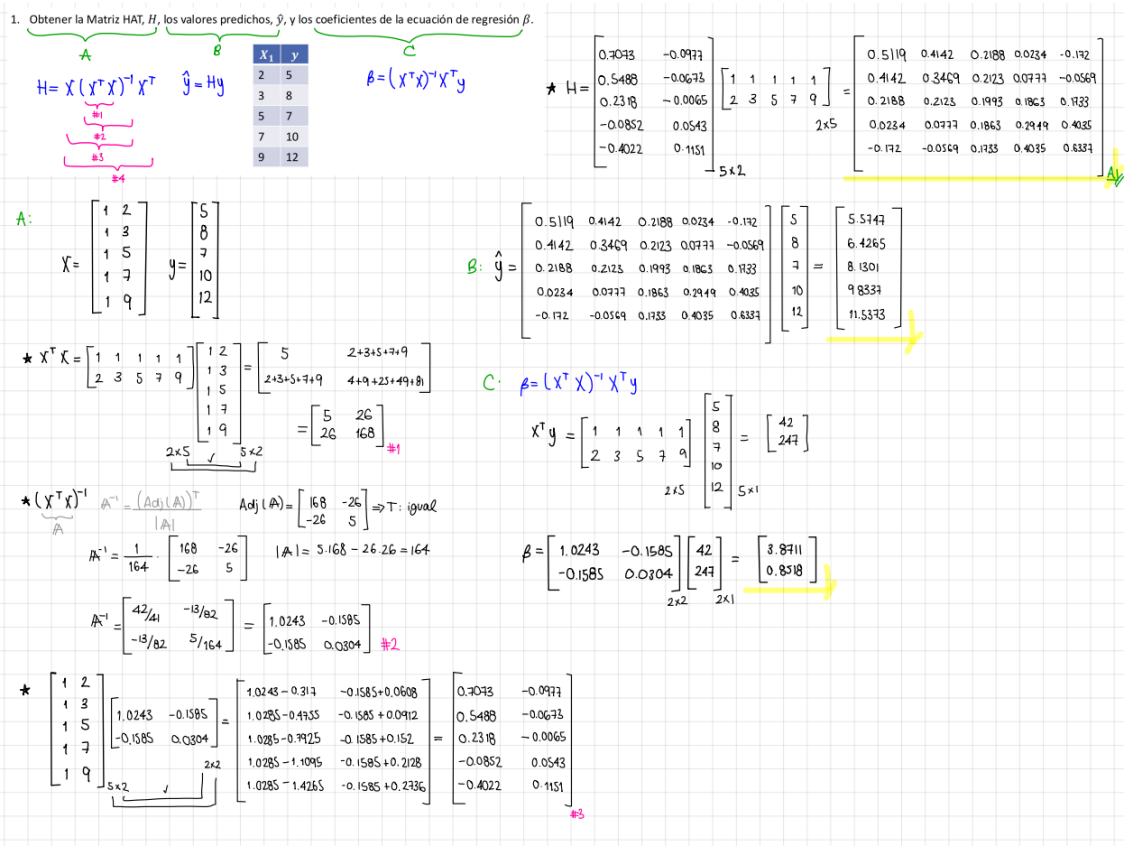
A01640495 | Carlos Alberto

A01285158 | Grace Aviance

1.Para el siguiente conjunto de datos, obtener la Matriz HAT H, los valores predichos, $\hat{y}$, y los coeficientes de la ecuación de regresión β.

[1]:
```
from IPython.display import Image
Image('Tarea1_EJ 1.png')
```

[1]:



2.Verifiar los resultados utilizando Python. Obtención de la Matriz HAT :

[2]:
```
import numpy as np

# MATRIZ BRINDADA
original_matrix=np.array([[2,5],[3,8],[5,7],[7,10],[9,12]])
original_matrix
```

[2]:
```
array([[ 2,  5],
       [ 3,  8],
       [ 5,  7],
       [ 7, 10],
       [ 9, 12]])
```

[3]:
```
# NUESTRA MATRIZ X
x=original_matrix[:,0]
# Le agregamos una columna de unos
X = np.column_stack((np.ones(x.shape[0]), x))
X
```

[3]:
```
array([[1., 2.],
       [1., 3.],
       [1., 5.],
       [1., 7.],
       [1., 9.]])
```

[4]:
```
# CALCULAMOS (X^t . X)^inversa
XtX_inv = np.linalg.inv(np.dot(X.T, X))
XtX_inv
```

[4]:
```
array([[ 1.02439024, -0.15853659],
       [-0.15853659,  0.0304878 ]])
```

[5]:
```
# MATRIZ HAT
H = np.dot(np.dot(X, XtX_inv), X.T)

print("Matriz Hat (H):")
print(H)
```

```
Matriz Hat (H):
[[ 0.51219512  0.41463415  0.2195122   0.02439024 -0.17073171]
 [ 0.41463415  0.34756098  0.21341463  0.07926829 -0.05487805]
 [ 0.2195122   0.21341463  0.20121951  0.18902439  0.17682927]
 [ 0.02439024  0.07926829  0.18902439  0.29878049  0.40853659]
 [-0.17073171 -0.05487805  0.17682927  0.40853659  0.6402439 ]]
```

[6]:
```
# EXTRA: Valores de apalancamiento
leverage_values=np.diag(H)
print(leverage_values)
print(np.sum(leverage_values))
```

```
[0.51219512 0.34756098 0.20121951 0.29878049 0.6402439 ]
1.9999999999999998
```

Calculo de valores predichos:

[7]:
```
y=original_matrix[:,1]
y_hat = np.dot(H, y)
print(y_hat)
```

```
[ 5.6097561   6.48170732  8.22560976  9.9695122  11.71341463]
```

Coeficientes de la ecuación de regresión

[8]:
```
beta = np.dot(XtX_inv, np.dot(X.T, y))
beta
```

[8]:
```
array([3.86585366, 0.87195122])
```

3. Utilizando los Datos "Cirugía de Hígado" obtener la matriz HAT, los valores predichos, ŷ, y los coeficientes de la ecuación de regresión utilizando el método de matrices. (Puede realizarse con cualquier paquete).

[9]:
```
# Importamos librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn import linear_model
from sklearn.metrics import r2_score
import scipy.stats as stats
from sklearn.feature_selection import RFE
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
```

[10]:
```
dataframe=pd.read_csv('Tarea1_dataset.csv')
dataframe.head(3)
```

[10]:

	Factor Coagulación	Índice pronóstico	Función de enzima	Función de hígado	Edad	Género	Alcohol(moderado)	Alcohol(severo)	Sobrevivencia(días)
0	6.7	62	81	2.59	50	0	1	0	695
1	5.1	59	66	1.70	39	0	0	0	
2	7.4	57	83	2.16	55	0	0	0	

```
1                   403
2                   710
```

[11]:
```
dataframe.shape
```

[11]:
```
(108, 9)
```

[12]:
```
# Quito la columna 'Sobrevivencia(días)'
x_df = dataframe.drop('Sobrevivencia(días)', axis=1)
```

[13]:
```
# Variable dependiente
y_df=dataframe['Sobrevivencia(días)']
```

[14]:
```
# Selecciono las tres últimas columnas (que no se deben escalar)
columns_to_exclude = x_df.iloc[:, -3:].to_numpy()

# Selecciono las columnas que se van a escalar (excluyendo las tres últimas)
columns_to_scale = x_df.iloc[:, :-3].to_numpy()
y=y_df.to_numpy()
```

[15]:
```
# ESCALAMOS CON MINMAX
scaler=MinMaxScaler()

# Escalo las columnas seleccionadas
scaled_columns = scaler.fit_transform(columns_to_scale)

# Combino las columnas escaladas con las columnas no escaladas
x = np.hstack((scaled_columns, columns_to_exclude))
```

Matriz HAT

[16]:
```
X = np.column_stack((np.ones(x.shape[0]), x))
y = y
```

Obtención de los valores de la matriz HAT

[17]:
```
#Calculo matriz HAT
XtX_inv = np.linalg.inv(np.dot(X.T, X))
H = np.dot(np.dot(X, XtX_inv), X.T)
H
```

[17]:
```
array([[ 0.03714196,  0.01278513,  0.03153001, …, -0.0107616 ,
         0.03002929,  0.00244981],
       [ 0.01278513,  0.06106075,  0.04827415, …,  0.02342793,
         0.0143413 ,  0.03831749],
       [ 0.03153001,  0.04827415,  0.09511866, …,  0.02121118,
         0.02055136,  0.03079561],
       …,
       [-0.0107616 ,  0.02342793,  0.02121118, …,  0.10349403,
```

```
         0.00417997,  0.03037262],
       [ 0.03002929,  0.0143413 ,  0.02055136, …,  0.00417997,
         0.06246334, -0.00802438],
       [ 0.00244981,  0.03831749,  0.03079561, …,  0.03037262,
        -0.00802438,  0.05864247]])
```

[18]:
```
# NÚMERO DE OBSERVACIONES:
H.shape
```

[18]:
```
(108, 108)
```

Para calcular los valores predichos hago el producto punto de la matriz HAT previamente calculada con vector y de mi dataset original

[19]:
```
y_hat = np.dot(H, y)
y_hat
```

[19]:
```
array([ 706.25623722,  430.82292124,  732.22983195,  425.03957611,
       1454.587552  ,  317.98512422,  561.43434506,  620.57029786,
        814.99238167,  761.68391557,  989.9519364 ,  228.02181205,
       1394.13461057, 1050.29172024,  937.95668893,  732.48434715,
        418.09976441,   58.32293366,  857.99665125,  939.03472439,
        555.53153245,  366.02513011,  485.52130235,  817.77817446,
        861.31877768,  669.35808718,  592.75876392, 1619.88673751,
        399.48541635,  596.03964243,  257.21831085,  163.93595245,
        594.67195134, 1143.59947369,  512.12738287,  702.86128551,
        629.09497899,  563.49249113,  556.62971999,  607.66774824,
        423.46168884,  569.50576666, 1363.25897377,  625.40911054,
        688.74643782,  590.69719298,  948.046226  , 1333.78069235,
        712.00261732, 1065.38071841,  462.70846831,  649.64082214,
        690.30288166,  958.05440017,  330.98090005,  811.81615789,
        635.24982009,  367.64186795,  629.61451147,  469.62155034,
        154.73023565,  369.81392347,  403.26975861,  902.08295831,
        600.59893803,  772.29123143,  207.7278467 ,  396.43186123,
       1229.83955625,  633.00436345,  424.81983957,  475.31740512,
        993.2862035 ,  197.38922035, 1290.08457201,  605.07146825,
       1006.51638101,  817.39664738,  616.69071501,  333.13872555,
        621.92856032,  678.01871248,  455.39524007,  842.81200029,
        215.61353579, 1044.1331181 ,  629.57478633,  526.79553852,
        902.07905149,  722.37107654,  212.15682923,  745.77909508,
        242.52437636,  565.21119037,  222.12633537,  839.18478601,
        535.39633923,  868.71602558,  607.10394711,  277.08092447,
       1086.53421927,  698.14563064, 1093.148018  ,  566.51505469,
        568.33491022,  584.36793863,  385.90989128,  459.71998145])
```

Para calcular los coeficientes de la ecuación de regresión:

[20]:
```
beta = np.dot(XtX_inv, np.dot(X.T, y))
print(f"Coeficientes del modelo: {beta}")
```

```
Coeficientes del modelo: [-575.86374747  453.26085291  738.23244281
854.14304366  429.43164879
   25.63892773   13.09258075  -41.26764482  195.70703222]
```

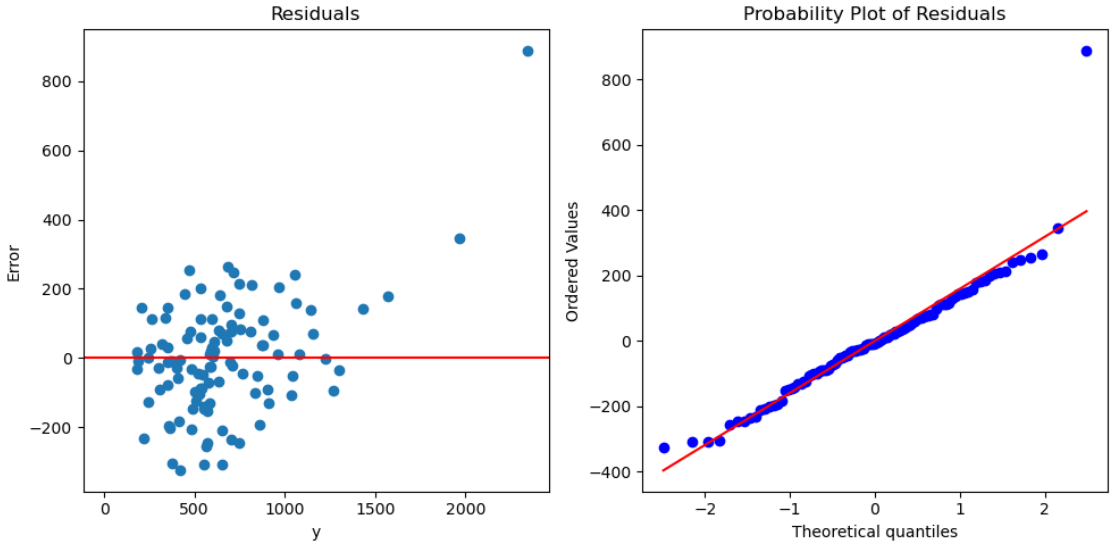
[21]:
```
residuals = y - y_hat
residuals
```

[21]:
```
array([ -11.25623722,  -27.82292124,  -22.22983195,  -76.03957611,
        888.412448  ,   30.01487578,  -43.43434506,  128.42970214,
        241.00761833,  206.31608443, -244.9519364 ,   28.97818795,
        178.86538943, -192.29172024, -235.95668893,   76.51565285,
        263.90023559,  146.67706634, -307.99665125, -101.03472439,
       -196.53153245,  -13.02513011,  113.47869765, -255.77817446,
       -210.31877768,   81.64191282,  -47.75876392,  345.11326249,
         77.51458365,    3.96035757,  185.78168915,   17.06404755,
       -183.67195134, -106.59947369,  -30.12738287,  -68.86128551,
         48.90502101, -201.49249113,   80.37028001,   97.33225176,
        112.53831116,   12.49423334,  -93.25897377,  -87.40911054,
       -206.74643782,   20.30280702,   11.953774  ,  -33.78069235,
       -131.00261732,   12.61928159,  -57.70846831,  -70.64082214,
       -140.30288166, -307.05440017,  -28.98090005,  -44.81615789,
       -148.24982009, -125.64186795,   75.38548853,  246.37844966,
        111.26976435,   -8.81392347,   56.73024139,  157.91704169,
        -98.59893803,  109.70876857,  144.2721533 ,  -89.43186123,
         -2.83955625, -125.00436345,   -5.81983957,   60.68259488,
        -91.2862035 ,   -8.38922035,  142.91542799,  209.92853175,
        137.48361899, -246.39664738,  -25.69071501,  199.86127445,
        -87.92856032, -304.01871248, -233.39524007,   38.18799971,
        254.38646421, -131.1331181 , -102.57478633,  149.20446148,
        -52.07905149, -153.37107654,  -30.15682923, -324.77909508,
          2.47562364,   45.78880963,  115.87366463,   35.81521399,
        214.60366077,   66.28397442,  -24.10394711,   41.91907553,
         71.46578073, -145.14563064,  -52.148018  ,   22.48494531,
         30.66508978,   70.63206137,   -8.90989128,  182.28001855])
```

EXTRA:

[22]:
```
import matplotlib.pyplot as plt
import scipy.stats as stats

# Crear la figura con dos subplots en una fila y dos columnas
fig, axs = plt.subplots(1, 2, figsize=(10, 5))  # figsize ajusta el tamaño␣
 ↪total de los gráficos

# Graficar los residuos
```

```
axs[0].scatter(y, residuals)
axs[0].axline((0, 0), slope=0, color='red')
axs[0].set_xlabel('y')
axs[0].set_ylabel('Error')
axs[0].set_title('Residuals')

# Q-Q plot para los residuos
stats.probplot(residuals, dist="norm", plot=axs[1])
axs[1].set_title('Probability Plot of Residuals')

# Ajustar el layout para que no se superpongan los gráficos
plt.tight_layout()

# Mostrar la figura con ambos gráficos
plt.show()
```



4. Concluir sobre el significado de los valores de apalancamiento y formular la ecuación de regresión.

[23]:
```
leverage_values=np.diag(H)
print("Elementos de la diagonal:")
print(leverage_values)
print(f"TRAZA: {np.sum(leverage_values)}")
```

```
Elementos de la diagonal:
[0.03714196 0.06106075 0.09511866 0.07234516 0.11942645 0.08065189
 0.07642418 0.07843142 0.05461582 0.07009281 0.07159644 0.04962392
 0.13226894 0.05700003 0.07881398 0.07780303 0.11185145 0.10082189
 0.05958639 0.08650172 0.06043351 0.11175274 0.12660129 0.03285494
 0.06822654 0.05336978 0.06117105 0.21015924 0.0637799  0.04627201
```

```
 0.07765548 0.1382091  0.07017845 0.07616134 0.06096139 0.05026388
 0.09800896 0.2127648  0.03475134 0.08648634 0.05249874 0.14410901
 0.18602117 0.03727933 0.12612543 0.05678346 0.06061771 0.12127331
 0.05897746 0.13267214 0.05612883 0.12188549 0.05172967 0.11945507
 0.08982727 0.06521368 0.15044357 0.08925879 0.09766654 0.1493418
 0.09388399 0.10980073 0.05614709 0.0760336  0.04568778 0.05055969
 0.11489641 0.05145828 0.10340551 0.06096153 0.05972407 0.06740464
 0.09018131 0.10956971 0.11351692 0.09159203 0.05941746 0.05413813
 0.08807509 0.12946688 0.03965555 0.09675421 0.08221149 0.05637406
 0.09542024 0.10265085 0.10503717 0.08818435 0.0813272  0.03353834
 0.04151066 0.08399356 0.05901144 0.08170849 0.10319491 0.05440942
 0.06707476 0.09243314 0.05273941 0.05580399 0.07672453 0.08439488
 0.12997958 0.03064927 0.07615033 0.10349403 0.06246334 0.05864247]
TRAZA: 8.999999999999975
```

Dado que la traza de la matriz H es 8.99 osea p=9, significa que este modelo debería de tener 9 parámetros, incluyendo la intersección.

Los elementos de la diagonal de H , son los puntos de apalancamiento que muestran la influencia que tiene cada observación sobre la línea de regresión. Para considerar un valor de h_{ii} como grande es comparar h_{ii} con el valor promedio de los apalancamientos. El promedio de los valores de h_{ii} es: $\frac{p}{n}$

El número total de observaciones es n=108. Un valor de apalancamiento significativamente mayor que p/n se considera elevado. En este caso, $\frac{p}{n} = 0.083$

Formulación de la ecuación de regresión= -575.86374747 β_0 + 453.26085291 β_1 + 738.23244281 β_2 + 854.14304366 β_3 + 429.43164879 β_4 + 25.63892773 β_5 + 13.09258075 β_6 - 41.26764482 β_7 + 195.70703222 β_8