Formulario 8: Diseño Factorial 2^k

Diseño de experimentos

Andrés Felipe Pico Zúñiga

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import seaborn as sns
import statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
import statsmodels.formula.api as smf
import scipy.stats as stats
```

Enunciado

En una planta donde se fabrican semiconductores se quiere mejorar el rendimiento del proceso vía diseño de experimentos. De acuerdo con la experiencia del grupo de mejora, los factores que podrían tener mayor influencia sobre la variable de respuesta (rendimiento), así como los niveles de prueba utilizados son los siguientes:

- A = Nivel de la abertura (pequeña, grande) = (-1,1).
- B = Tiempo de exposición (20% abajo, 20% arriba)= (-1,1).
- C = Tiempo de revelado (30 seg, 45 seg)= (-1,1).
- D = Dimensión de la máscara (pequeña, grande)= (-1,1).
- E = Tiempo de grabado (14.5 min, 15.5 min)= (-1,1).

Se decide correr un experimento 25 con una sola corrida o réplica para estudiar estos cinco factores. Se hacen las 32 corridas a nivel proceso.

Creación y visualización del dataframe

```
A = Nivel_abertura =  [ "-1" ,  "1" ]
B = Tiempo_exposición  =  [ "-1" ,  "1" ]
C = Tiempo_revelado  =  [ "-1" ,  "1" ]
D = Dimensión_máscara  =  [ "-1" ,  "1" ]
E = Tiempo_grabado  =  [ "-1" ,  "1" ]
rendimiento = [7, 9, 34, 55, 16, 20, 40, 60,
               8, 10, 32, 50, 18, 21, 44, 61,
               18, 12, 35, 52, 15, 22, 45, 65,
               6, 10, 30, 53, 15, 20, 41, 63]
```

```
Nivel_abertura =  (["-1"] * 1 + ["1"] * 1)*16

Tiempo_exposición = (["-1"] * 2 + ["1"] * 2 ) * 8
Tiempo_revelado = (["-1"] * 4 + ["1"] * 4)*4

Dimensión_máscara = (["-1"] * 8 + ["1"] * 8 ) * 2
Tiempo_grabado = ["-1"] * 16 + ["1"] * 16

df = pd.DataFrame({
  'A': Nivel_abertura,
  'B': Tiempo_exposición,
  'C': Tiempo_revelado,
  'D': Dimensión_máscara,
  'E': Tiempo_grabado,
  'rendimiento': rendimiento
})

print(df)
```

```
##       A   B   C   D   E  rendimiento
## 0    -1  -1  -1  -1  -1            7
## 1     1  -1  -1  -1  -1            9
## 2    -1   1  -1  -1  -1           34
## 3     1   1  -1  -1  -1           55
## 4    -1  -1   1  -1  -1           16
## 5     1  -1   1  -1  -1           20
## 6    -1   1   1  -1  -1           40
## 7     1   1   1  -1  -1           60
## 8    -1  -1  -1   1  -1            8
## 9     1  -1  -1   1  -1           10
## 10   -1   1  -1   1  -1           32
## 11    1   1  -1   1  -1           50
## 12   -1  -1   1   1  -1           18
## 13    1  -1   1   1  -1           21
## 14   -1   1   1   1  -1           44
## 15    1   1   1   1  -1           61
## 16   -1  -1  -1  -1   1           18
## 17    1  -1  -1  -1   1           12
## 18   -1   1  -1  -1   1           35
## 19    1   1  -1  -1   1           52
## 20   -1  -1   1  -1   1           15
## 21    1  -1   1  -1   1           22
## 22   -1   1   1  -1   1           45
## 23    1   1   1  -1   1           65
## 24   -1  -1  -1   1   1            6
## 25    1  -1  -1   1   1           10
## 26   -1   1  -1   1   1           30
## 27    1   1  -1   1   1           53
## 28   -1  -1   1   1   1           15
## 29    1  -1   1   1   1           20
## 30   -1   1   1   1   1           41
## 31    1   1   1   1   1           63
```

Tabla ANOVA

Efectos principales e interacciones dobles

```
modelo1 = ols(
  "rendimiento ~ (A+B+C+D+E)**2",
  data=df
).fit()
anova_result = sm.stats.anova_lm(modelo1, typ=1)
print (anova_result)
```

```
##              df      sum_sq      mean_sq            F        PR(>F)
## A           1.0  1001.28125  1001.281250   156.679707  1.113803e-09
## B           1.0  8877.78125  8877.781250  1389.188264  5.576883e-17
## C           1.0   657.03125   657.031250   102.811736  2.264156e-08
## D           1.0    16.53125    16.531250     2.586797  1.273092e-01
## E           1.0     9.03125     9.031250     1.413203  2.518705e-01
## A:B         1.0   586.53125   586.531250    91.779951  4.971986e-08
## A:C         1.0     9.03125     9.031250     1.413203  2.518705e-01
## A:D         1.0     2.53125     2.531250     0.396088  5.379999e-01
## A:E         1.0     0.78125     0.781250     0.122249  7.311672e-01
## B:C         1.0     3.78125     3.781250     0.591687  4.529740e-01
## B:D         1.0     0.03125     0.031250     0.004890  9.451172e-01
## B:E         1.0     0.03125     0.031250     0.004890  9.451172e-01
## C:D         1.0    16.53125    16.531250     2.586797  1.273092e-01
## C:E         1.0     0.78125     0.781250     0.122249  7.311672e-01
## D:E         1.0    26.28125    26.281250     4.112469  5.955770e-02
## Residual   16.0   102.25000     6.390625          NaN           NaN
```

```
modelo2 = sm.OLS.from_formula(
  'rendimiento ~ (A+B+C+D+E)**2',
  data=df
).fit()
print(modelo2.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:            rendimiento   R-squared:                       0.991
## Model:                            OLS   Adj. R-squared:                  0.982
## Method:                 Least Squares   F-statistic:                     116.9
## Date:                vie, 17 oct 2025   Prob (F-statistic):           1.88e-13
## Time:                        12:55:36   Log-Likelihood:                -63.993
## No. Observations:                  32   AIC:                             160.0
## Df Residuals:                      16   BIC:                             183.4
## Df Model:                          15
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## Intercept      9.4375      1.788      5.280      0.000       5.648      13.227
## A[T.1]         0.6875      1.999      0.344      0.735      -3.549       4.924
## B[T.1]        24.1875      1.999     12.103      0.000      19.951      28.424
## C[T.1]         6.1875      1.999      3.096      0.007       1.951      10.424
## D[T.1]        -1.5625      1.999     -0.782      0.446      -5.799       2.674
## E[T.1]         2.9375      1.999      1.470      0.161      -1.299       7.174
```

```
## A[T.1]:B[T.1]    17.1250      1.788      9.580      0.000      13.336      20.914
## A[T.1]:C[T.1]     2.1250      1.788      1.189      0.252      -1.664       5.914
## A[T.1]:D[T.1]     1.1250      1.788      0.629      0.538      -2.664       4.914
## A[T.1]:E[T.1]     0.6250      1.788      0.350      0.731      -3.164       4.414
## B[T.1]:C[T.1]     1.3750      1.788      0.769      0.453      -2.414       5.164
## B[T.1]:D[T.1]    -0.1250      1.788     -0.070      0.945      -3.914       3.664
## B[T.1]:E[T.1]    -0.1250      1.788     -0.070      0.945      -3.914       3.664
## C[T.1]:D[T.1]     2.8750      1.788      1.608      0.127      -0.914       6.664
## C[T.1]:E[T.1]    -0.6250      1.788     -0.350      0.731      -4.414       3.164
## D[T.1]:E[T.1]    -3.6250      1.788     -2.028      0.060      -7.414       0.164
## ==============================================================================
## Omnibus:                        8.034   Durbin-Watson:                   2.194
## Prob(Omnibus):                  0.018   Jarque-Bera (JB):                6.651
## Skew:                           0.856   Prob(JB):                       0.0360
## Kurtosis:                       4.434   Cond. No.                         14.7
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Optimización

```
from scipy.optimize import minimize

# Define la función que deseas maximizar se antepone el signo (-)
def funcion(x):
    return -(9.437 + 0.687*x[0] + 24.187*x[1] + 6.187*x[2] - 1.563*x[3] + 2.937*x[4] + 17.125*x[0]*x[1]

# Define las restricciones para x
def variables(x):
    return x[0], x[1], x[2], x[3], x[4]

# Definir las restricciones de límite para x
restricciones = [(-1, 1), (-1, 1), (-1, 1), (-1, 1), (-1, 1)]

# Suprimir la salida de la optimización
res = minimize(funcion, [0, 0, 0, 0, 0], method='SLSQP', constraints={'type':'ineq', 'fun': variables},

# Imprimir el resultado
print("Resultado óptimo:")
```

```
## Resultado óptimo:
```

```
print("x:", res.x)
```

```
## x: [9.99999998e-01 9.99999997e-01 9.99999998e-01 1.27861136e-11
##  9.99999998e-01]
```

```
print("Valor Máximo:", -res.fun)  # Como minimizamos el negativo de la función
```

```
## Valor Máximo: 63.93499981295733
```

Colapsando el Factor E

```
modelo3 = ols(
  "rendimiento ~ (A+B+C+D)**2",
  data=df
).fit()
anova_result = sm.stats.anova_lm(modelo3, typ=1)
print (anova_result)
```

```
##              df      sum_sq      mean_sq            F        PR(>F)
## A           1.0  1001.28125  1001.281250   151.102852  4.665074e-11
## B           1.0  8877.78125  8877.781250  1339.741523  1.649690e-20
## C           1.0   657.03125   657.031250    99.152257  2.087253e-09
## D           1.0    16.53125    16.531250     2.494723  1.291741e-01
## A:B         1.0   586.53125   586.531250    88.513137  5.577236e-09
## A:C         1.0     9.03125     9.031250     1.362901  2.561204e-01
## A:D         1.0     2.53125     2.531250     0.381990  5.431858e-01
## B:C         1.0     3.78125     3.781250     0.570627  4.583996e-01
## B:D         1.0     0.03125     0.031250     0.004716  9.458999e-01
## C:D         1.0    16.53125    16.531250     2.494723  1.291741e-01
## Residual   21.0   139.15625     6.626488          NaN           NaN
```

Optimización

```
modelo3.params
```

```
## Intercept        10.90625
## A[T.1]            1.00000
## B[T.1]           24.12500
## C[T.1]            5.87500
## D[T.1]           -3.37500
## A[T.1]:B[T.1]    17.12500
## A[T.1]:C[T.1]     2.12500
## A[T.1]:D[T.1]     1.12500
## B[T.1]:C[T.1]     1.37500
## B[T.1]:D[T.1]    -0.12500
## C[T.1]:D[T.1]     2.87500
## dtype: float64
```

Colapsando el Factor D

```
modelo4 = ols(
  "rendimiento ~ (A+B+C)**2",
  data=df
).fit()
anova_result = sm.stats.anova_lm(modelo4, typ=1)
print (anova_result)
```

```
##              df      sum_sq     mean_sq            F        PR(>F)
## A           1.0  1001.28125  1001.28125   143.219203  7.613839e-12
## B           1.0  8877.78125  8877.78125  1269.841766  5.944113e-23
## C           1.0   657.03125   657.03125    93.979081  5.977636e-10
## A:B         1.0   586.53125   586.53125    83.895047  1.829092e-09
## A:C         1.0     9.03125     9.03125     1.291793  2.664954e-01
## B:C         1.0     3.78125     3.78125     0.540855  4.689226e-01
## Residual   25.0   174.78125     6.99125          NaN           NaN
```

Optimización

```
modelo4.params
```

```
## Intercept         9.21875
## A[T.1]            1.56250
## B[T.1]           24.06250
## C[T.1]            7.31250
## A[T.1]:B[T.1]    17.12500
## A[T.1]:C[T.1]     2.12500
## B[T.1]:C[T.1]     1.37500
## dtype: float64
```