SE LEEN LOS DATOS Y SE REALIZAN LAS ADAPTACIONES PARA EL ANALISIS

In [ ]:
```
import pandas as pd             # Manejo de datasets
import matplotlib.pyplot as plt # Manejo de gráficos
import seaborn as sns           # Manejo de gráficos con datasets
import numpy as np              # Matematicas y datasets

import numpy
from sklearn import linear_model
import statsmodels.api as sm

# Acceso al dataset

ruta = "./DATOS-RED-COMPLETOS-14-05.csv"
data = pd.read_csv(ruta)

# Verificar que no hay valores nulos en el data set
print ("Verificar que no hay valores nulos en el data set")
print()
print(data.isnull().sum())

# informacion sobre las características

print()
print ("informacion sobre las características del dataset")
print()

print(data.info())

# Informacion sobre las columnas

print()
print ("informacion sobre las columnas numericas del dataset")
print()

info_dataset=data.describe(exclude=object)
print(info_dataset)

# Se genera archivo excel para documentar

writer = pd.ExcelWriter('./info_dataset.xlsx', engine='openpyxl')
info_dataset.to_excel(writer, sheet_name='Numerico', index=True)

print()
print ("informacion sobre las columnas categoricas del dataset")
print()

info_dataset=data.describe(include=object)
print(info_dataset)
info_dataset.to_excel(writer, sheet_name='Categorico', index=True)
# Cerrar el archivo Excel
writer.close()
```

In [2]:
```
# Extraccion de informacion de las variable categoricas

lista = ['CODESPECIALIDAD', 'NIVEL', 'FAMILIA', 'AREA', 'SITUACION_GENERAL',
       'SITUACION', 'SEXO',  'Año_af', 'Edad', 'TIPOSELECCION',
       'RESULTADO_GENERAL', 'resultado', 'RES_ENTREVISTA',
        'ESPECIALIDAD-REQUERIDA',
        'DISCAPACIDAD',  'PLD',
        'TITULACION',  'ENTRE-30-50',
        'INTERMEDIA',
       'CUALIFICACION',  'RSB',  'PLD_MAYOR',
        'VCOBRAPRESTACION', 'VACCIONESCURSADAS',
       'VACCIONESSUPERADAS', 'LOCALIDAD CENTRO', ]

# Se guarda en el excel
```

```
writer = pd.ExcelWriter('./var-cate.xlsx', engine='openpyxl')

for x in lista:
   info_dataset=data.groupby(by=x)[x].count()
   info_dataset.to_excel(writer, sheet_name=x, index=True)

writer.close()
```

In [ ]:
```
# Se escala la columna Año accion formativa entre 0 y 1 por si se necesita mas adelante


new_min, new_max = 0, 1
old_min, old_max = data['Año_af'].min(), data['Año_af'].max()
data['Año_af'] = (data['Año_af'] - old_min) / (old_max - old_min) *
                                    (new_max - new_min) + new_min

# Se escala la columna horas totales

new_min, new_max = 0, 1
old_min, old_max = data['Horas_totales'].min(), data['Horas_totales'].max()
data['Horas_totales'] = (data['Horas_totales'] - old_min) /
                  (old_max - old_min) * (new_max - new_min) + new_min

# Definir tramos de edad por si se necesitan

bins = [0, 20, 30,40, 50, 60,70, float('inf')]
labels = ['0_20','21_30','31_40','41_50','51_60', '61_70', '>70']

# Aplicar la función cut para crear la nueva columna de tramos

data['tramo_edad'] = pd.cut(data['Edad'], bins=bins, labels=labels, right=False)


# Definir otro tramo de edad por si se necesitan

bins = [0, 30, float('inf')]
labels = ['0_30','mas30']

data['tramo_edad2'] = pd.cut(data['Edad'], bins=bins, labels=labels, right=False)


# cambiar los valores de una column por otros sin caracter de -


data['RESULTADO_GENERAL'] = data['RESULTADO_GENERAL'].str.replace("CUALIFICA-TOTAL-PARCIAL",
                                                                  "CUALIFICA_TOTAL_PARCIAL",
                                                                  case=False, regex=False)

data = data.rename(columns={'ESPECIALIDAD-REQUERIDA': 'ESPECIALIDAD_REQUERIDA'})

numerical_cols =['NIVEL',
                 'Año_af',
                 'Edad', 'Horas_totales',
                 'PTOS_ENTREVISTA', 'PTOS_ESPECIALIDAD',
                 'PTOS_DISCAPACIDAD', 'PTOS_PLD', 'PTOS_TITULACION',
                 'PTOS_EDAD_30_50', 'PTOS_SERVICIO_111',
                 'PTOS_CUALIFICACION', 'PTOS_RSB', 'PTOS_PLD_MAYORES',
                 'PUNTOS_TOTAL','CENSO']

# Graficando las columnas numericas

data[numerical_cols].hist(figsize=(10, 10), bins=20, color='blue', edgecolor='black')
plt.show()
```

In [ ]:
```
#Graficos extendidos

no_numerical_cols_l = ['CODESPECIALIDAD',
                       ]
```

```
data1=data.sort_values('CODESPECIALIDAD')
for co in no_numerical_cols_l:
   fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(5,30))
   sns.countplot(y=co, data=data1, ax=ax)
   ax.set_title(co)
   sns.set_theme(style="whitegrid",font_scale=0.75)
   sns.axes_style("darkgrid")
   plt.show()

no_numerical_cols_l = [
                    'AREA'
                    ]

for co in no_numerical_cols_l:
   fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(5,20))
   sns.countplot(y=co, data=data1, ax=ax)
   ax.set_title(co)
   sns.set_theme(style="whitegrid",font_scale=0.75)
   sns.axes_style("darkgrid")
   plt.show()


   no_numerical_cols_l = [
                    'FAMILIA'
                    ]

for co in no_numerical_cols_l:
   fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(5,10))
   sns.countplot(y=co, data=data1, ax=ax)
   ax.set_title(co)
   sns.set_theme(style="whitegrid",font_scale=0.75)
   sns.axes_style("darkgrid")
   plt.show()

no_numerical_cols = [
                    'SITUACION_GENERAL',
                    'SITUACION',
                    'SEXO',
                    'TIPOSELECCION',
                    'RESULTADO_GENERAL',
                    'resultado',
                    'RES_ENTREVISTA',
                    'ESPECIALIDAD_REQUERIDA',
                    'DISCAPACIDAD',
                    'PLD',
                    'TITULACION',
                    'ENTRE-30-50',
                    'INTERMEDIA',
                    'CUALIFICACION',
                    'RSB',
                    'PLD_MAYOR',
                    'VCOBRAPRESTACION',
                    'VACCIONESCURSADAS',
                    'VACCIONESSUPERADAS',
                    'LOCALIDAD CENTRO',
                    'tramo_edad'
                    ]


fig, ax = plt.subplots(nrows=len(no_numerical_cols), ncols=1, figsize=(6,100))
fig.subplots_adjust(hspace=0.5)

for i, col in enumerate(no_numerical_cols):
  sns.countplot(x=col, data=data, ax=ax[i])
  ax[i].set_title(col)
  ax[i].set_xticklabels(ax[i].get_xticklabels(),rotation=60)
sns.set_theme(style="whitegrid")
sns.axes_style("darkgrid")
plt.show()
```

CONVIRTIENDO VARIABLES CATEGORICAS EN BINARIAS

In [ ]:
```
#Convirtiendo variables categoricas

# Nos quedamos con las caracteristicas que nos interesan

temp_df = data[[
                'NIVEL',
                'FAMILIA',
                'SITUACION_GENERAL',
                'SEXO',
                'Edad',
                'tramo_edad',
                'tramo_edad2',
                'RESULTADO_GENERAL',
                'Horas_totales',
                'PTOS_ESPECIALIDAD',
                'ESPECIALIDAD_REQUERIDA',
                'PTOS_DISCAPACIDAD',
                'DISCAPACIDAD',
                'PTOS_PLD',
                'PLD',
                'PTOS_TITULACION',
                'TITULACION',
                'PTOS_SERVICIO_111',
                'INTERMEDIA',
                'PTOS_CUALIFICACION',
                'CUALIFICACION',
                'PTOS_RSB',
                'RSB',
                'PUNTOS_TOTAL',
                'VCOBRAPRESTACION',
                'VACCIONESCURSADAS',
                'VACCIONESSUPERADAS',
                'Año_af',
                'LOCALIDAD CENTRO'
            ]]

temp_df = pd.get_dummies(temp_df,columns=[
       'NIVEL',
       'FAMILIA',
       'SITUACION_GENERAL',
       'SEXO',
       'tramo_edad',
       'tramo_edad2',
       'RESULTADO_GENERAL',
       'ESPECIALIDAD_REQUERIDA',
       'DISCAPACIDAD',
       'PLD',
       'TITULACION',
       'INTERMEDIA',
       'CUALIFICACION',
       'RSB',
       'VCOBRAPRESTACION',
       'VACCIONESCURSADAS',
       'VACCIONESSUPERADAS',
       'LOCALIDAD CENTRO'
       ],
        dtype=int,drop_first=True)

# preparacion de datos

temp_df['const_'] = 1
temp_df.columns
temp_df
```

VARIABLES

In [6]:
```
X = ['SEXO_MUJER',
   # 'tramo_edad2_mas30', # Característica a analisis (T)
     'DISCAPACIDAD_SI',
     'PLD_SI',
     'ESPECIALIDAD_REQUERIDA_SI',
     'INTERMEDIA_SI',
     'CUALIFICACION_SI',
     'TITULACION_SI',
     'RSB_SI',
     'VACCIONESCURSADAS_SI',
     'VACCIONESSUPERADAS_SI'
     ]

REGRE = ['SEXO_MUJER',
     'tramo_edad2_mas30',
     'DISCAPACIDAD_SI',
     'PLD_SI',
     'TITULACION_SI',
     'INTERMEDIA_SI',
     'CUALIFICACION_SI',
     'RSB_SI',
     'VACCIONESCURSADAS_SI',
     'VACCIONESSUPERADAS_SI',
     'ESPECIALIDAD_REQUERIDA_SI' ,
     'const_'
     ]

T = 'tramo_edad2_mas30' # Tratamiento

Y= 'RESULTADO_GENERAL_CUALIFICA_TOTAL_PARCIAL' # Variable dependiente

categoricas=[
     'SEXO_MUJER',
     'tramo_edad2_mas30'
     'DISCAPACIDAD_SI',
     'PLD_SI',
     'TITULACION_SI',
     'INTERMEDIA_SI',
     'CUALIFICACION_SI',
     'RSB_SI',
     'VACCIONESCURSADAS_SI',
     'VACCIONESSUPERADAS_SI',
     'ESPECIALIDAD_REQUERIDA_SI',
     'RESULTADO_GENERAL_CUALIFICA_TOTAL_PARCIAL']
```

Division en tratamiento y control

In [ ]:
```
# DIVISION EN TRATAMIENTO Y CONTROL

t_data = temp_df.loc[temp_df.tramo_edad2_mas30==1]

c_data = temp_df.loc[temp_df.tramo_edad2_mas30==0]


# Ver cual es la diferencia de medias de la variable objetivo entre tratamiento y control

treated_mean = t_data[Y].mean()
contol_mean =  c_data[Y].mean()
treatment_effect = treated_mean - contol_mean
print(" Media tratado: ",treated_mean, " Media control:  ", contol_mean ,
       " Efecto: ",treatment_effect)

# Tratamiento

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(3,3))
sns.countplot(x=Y, data=t_data, ax=ax)
ax.set_title('Tratamiento')
sns.set_theme(style="whitegrid",font_scale=0.5)
sns.axes_style("darkgrid")
```

```
plt.show()

# Control

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(3,3))
sns.countplot(x=Y, data=c_data, ax=ax)
ax.set_title('Control')
sns.set_theme(style="whitegrid",font_scale=0.5)
sns.axes_style("darkgrid")
plt.show()

# REGRESIÓN

pre_treatment_vars = [
   'tramo_edad2_mas30' ,
   'const_'
]

X1 = temp_df[pre_treatment_vars]
Y1 = temp_df[Y]

reg = sm.OLS(Y1,X1)
res = reg.fit()
print(res.summary())
```

SIMILITUDES ANTES DEL PAREADO

In [ ]:
```
# Una forma estándar de comprobar la similitud de grupos en
# experimentos es realizar una prueba estadística comparando
# las medias o proporciones entre ambos grupos.
# La hipótesis nula en todas nuestras pruebas establece que
# las medias o proporciones entre los grupos de control y de
# tratamiento son iguales. Por lo tanto, un valor p bajo nos
# llevará a rechazar dicha hipótesis y a concluir que existen
# diferencias entre los grupos.


from scipy.stats import ttest_ind

from statsmodels.stats.proportion import proportions_ztest

def perform_ztest(count1, nobs1, count2, nobs2):
    z_stat, p_val = proportions_ztest([count1, count2], [nobs1, nobs2])
    return z_stat, p_val

ztest_results = []
for var in X:
    count_treated = t_data[var].sum()
    count_control = c_data[var].sum()
    nobs_treated = t_data[var].count()
    nobs_control = c_data[var].count()
    z_stat, p_val = perform_ztest(count_treated, nobs_treated,
                                  count_control, nobs_control)
    ztest_results.append({'Variable': var, 'Z-Statistic': round(z_stat, 2),
                          'P-Value': f"{p_val:.3f}"})

ztest_results = pd.DataFrame(ztest_results)

ztest_results
```

MATCHING EXACTO

In [ ]:
```
t_data_i = temp_df.loc[temp_df.tramo_edad2_mas30==1]
c_data_i = temp_df.loc[temp_df.tramo_edad2_mas30==0]

t_data_i_a = t_data_i[X]
c_data_i_a = c_data_i[X]

indices=pd.DataFrame({"i": [0], "j": [0]}, index=[0])
```

```
lista_usados=[]

for i in range(0, len(t_data_i_a)-1):
   filat = t_data_i_a.iloc[i]
   encontrado = False
   j=0
   while (encontrado ==False) and (j <= (len(c_data_i_a)-1)):
      if (j not in lista_usados): # No ha sido utilizado todavía
         filac = c_data_i_a.iloc[j]
         if filat.equals(filac):
            encontrado=True
            lista_usados.append(j)
            d1 = {"i": [i], "j": [j]}
            df1 = pd.DataFrame(d1, index=[i])
            indices=pd.concat([indices, df1])
            print("Par ({i}, {j})",i,j)
      j=j+1
print(indices)

matched_tratamiento_indices = indices['i']
matched_control_indices = indices['j']
new_tratamiento =t_data_i.iloc[matched_tratamiento_indices]
new_control =c_data_i.iloc[matched_control_indices]
matched_data = pd.concat([new_tratamiento, new_control])
matched_data
```

ANALISIS DE SIMILITUD DEL GRUPO PAREADO

In [ ]:
```
# Una forma estándar de comprobar la similitud de grupos en
# experimentos es realizar una prueba estadística comparando
# las medias o proporciones entre ambos grupos.
# La hipótesis nula en todas nuestras pruebas establece que
# las medias o proporciones entre los grupos de control y de
# tratamiento son iguales. Por lo tanto, un valor p bajo nos
# llevará a rechazar dicha hipótesis y a concluir que existen
# diferencias entre los grupos.


from scipy.stats import ttest_ind

matched_t = matched_data[matched_data[T] == 1]
matched_c = matched_data[matched_data[T] == 0]

from statsmodels.stats.proportion import proportions_ztest

def perform_ztest(count1, nobs1, count2, nobs2):
    z_stat, p_val = proportions_ztest([count1, count2], [nobs1, nobs2])
    return z_stat, p_val

ztest_results = []
for var in X:
    count_treated = matched_t[var].sum()
    count_control = matched_c[var].sum()
    nobs_treated = matched_t[var].count()
    nobs_control = matched_c[var].count()
    z_stat, p_val = perform_ztest(count_treated, nobs_treated,
                                  count_control, nobs_control)
    ztest_results.append({'Variable': var, 'Z-Statistic': round(z_stat, 2),
                          'P-Value': f"{p_val:.3f}"})

ztest_results = pd.DataFrame(ztest_results)

ztest_results
```

EFECTO SOBRE LOS TRATADOS (MEDIAS)

In [ ]:
```
lista_caracteristicas=[Y]
```

```
for car in lista_caracteristicas:
   treated_mean = matched_data[matched_data[T] == 1][car].mean()
   matched_contol_mean =  matched_data[matched_data[T] == 0][car].mean()
   treatment_effect = treated_mean - matched_contol_mean
   print(car," Media tratado: ",treated_mean, " MEdia control:  ",
         matched_contol_mean ,
          " Efecto: ",treatment_effect)
```

EFECTO ENTRE LOS TRATADOS, MEDIANTE REGRESIÓN LINEAL

In [ ]:

```
#Tratamiento

dfm1 = matched_data

X1 = dfm1[REGRE]
Y1 = dfm1[Y]

reg = sm.OLS(Y1,X1)
res = reg.fit()

print(res.summary())
```