Anexos

Anexo 1

A1.1. Obtención de los *DataFrame*

```
import pandas as pd
from sklearn import preprocessing
from pandas import ExcelWriter
import time
import statsmodels.api as sm
start_time = time.time()


# Importamos datos de las fases inicial y no inicial
Fase_Inicial = pd.read_excel('C:/Users/34630/Desktop/TFG ASP/qfaseini.xlsx')
Fase_No_Inicial = pd.read_excel('C:/Users/34630/Desktop/TFG ASP/qfasenoini.xlsx')
Selectivitat = pd.read_excel('C:/Users/34630/Desktop/TFG ASP/dadespersnomespreins.xlsx')
# Empezamos trabajando sobre Fase_Inicial
# Paso 1 - Comenzamos a depurar los datos
#1.1-Me quedo con los alumnos que hacen indus
DF1=Fase_Inicial[Fase_Inicial['CODI_PROGRAMA']==752]
#1.2- Ordeno por Curso y Cuatri
DF2=DF1.sort_values(['CODI_EXPEDIENT','CURS','QUAD'])
#1.3- Elimino alumnos que tienen Cuatri = 0 (Valor anormal)
DF3=DF2[DF2['QUAD']!=0]
#1.4- Quito las convalidaciones (En el grupo de clase están)
DF4=DF3[DF3['GRUP_CLASSE']!='CONV'].copy()
DF5=DF4.drop(DF4[(DF4['CURS'] == 2017) & (DF4['QUAD'] == 2)].index)
#1.5- Aplico la función nota, que conserva la nota final si su valor es igual o superior a 5, y asigna None en caso contrario
def nota(nota):
    result=None
```

```
    if nota >=5:

        result = nota

    return (result)
```

#1.6- Aplico la función a la columna 'NOTA_NUM_DEF' y creo una nueva columna 'NotaFinalFI'

```
DF5['NotaFinalFI']=DF5['NOTA_NUM_DEF'].apply((lambda x: nota(x)))
```

#1.7- Creo una tabla con los valores de la columna de 'NotaFinalFI', Esta tabla ya tiene únicamente los alumnos que han superado la Fase Inicial

```
Taula1=pd.pivot_table(DF5, values='NotaFinalFI', index='CODI_EXPEDIENT', columns='CODI_UPC_UD',aggfunc=max)

Taula1=Taula1.dropna().copy()

Taula1.rename(columns={240011: 'ALG', 240012: 'CALC1', 240013: 'MECFON', 240014: 'QUIM1', 240015: 'FONINFO', 240021: 'GEO', 240022: 'CALC2', 240023: 'TERMO', 240024: 'QUIM2', 240025: 'EXPRE'}, inplace=True)
```

#1.8- Creo una tabla con los valores de la nota media de cada alumno por cada asignatura

```
Taula2=pd.pivot_table(DF5, values='NOTA_NUM_DEF', index='CODI_EXPEDIENT', columns='CODI_UPC_UD', aggfunc='mean')

Taula2=Taula2.dropna().copy()

Taula2.rename(columns={240011: 'M-ALG', 240012: 'M-CALC1', 240013: 'M-MECFON', 240014: 'M-QUIM1', 240015: 'M-FONINFO', 240021: 'M-GEO', 240022: 'M-CALC2', 240023: 'M-TERMO', 240024: 'M-QUIM2', 240025: 'M-EXPRE'}, inplace=True)
```

#1.9- Creo una tabla con el número de convocatorias de cada alumno por cada asignatura

```
Taula3=pd.pivot_table(DF5, values='NOTA_NUM_DEF', index='CODI_EXPEDIENT', columns='CODI_UPC_UD', aggfunc='count')

Taula3=Taula3.dropna().copy()

Taula3.rename(columns={240011: 'C-ALG', 240012: 'C-CALC1', 240013: 'C-MECFON', 240014: 'C-QUIM1', 240015: 'C-FONINFO', 240021: 'C-GEO', 240022: 'C-CALC2', 240023: 'C-TERMO', 240024: 'C-QUIM2', 240025: 'C-EXPRE'}, inplace=True)

Data1_FaseIni=pd.merge(Taula1,Taula2,on='CODI_EXPEDIENT',how='inner')

Data1_FaseIni=Data1_FaseIni.dropna().copy()

Data2_FaseIni=pd.merge(Data1_FaseIni,Taula3,on='CODI_EXPEDIENT',how='inner')

Data2_FaseIni=Data2_FaseIni.dropna().copy()

# Empiezo con la selectividad

#Uno la tabla de selectividad con Data2_FaseIni para crear el DataFrame3 mas tarde

Data3_FaseIni=pd.merge(Data2_FaseIni,Selectivitat[['CODI_EXPEDIENT','NOTA_ACCES']],on='CODI_EXPEDIENT',how='inner')

# Empiezo con la fase no inicial

#Repito el Paso 1 que he utilizado anteriormente con la fase inicial

DF6=Fase_No_Inicial[Fase_No_Inicial['CODI_PROGRAMA']==752]
```

```
DF7=DF6.sort_values(['CODI_EXPEDIENT','CURS','QUAD'])

DF8=DF7[DF7['QUAD']!=0]

DF9=DF8[DF8['GRUP_CLASSE']!='CONV'].copy()

#Hago una lista con las asignaturas que quiero conservar al final

Q3 = ['240132','240133','240131','240033','240031','240032']

#Elimino las filas del fichero con asignaturas que no me interesan

DF10=DF9.drop(DF9[DF9.isin(Q3)['CODI_UPC_UD']==False].index)

#Cambio nombre de las asignaturas

DF10.CODI_UPC_UD= DF10.CODI_UPC_UD.replace({'240132': 'INFO', '240133': 'MEC','240131': 'EDOS', '240033': 'MATERIALS', '240031': 'ELECTRO', '240032': 'MÈTODES'})

#Creo una tabla con la primera nota de cada alumno en cada asignatura, es la única tablaque necesito de la faso no inicial

Data_FaseNoIni=pd.pivot_table(DF10, values='NOTA_NUM_DEF', index='CODI_EXPEDIENT', columns='CODI_UPC_UD', aggfunc='first')

#Uno las tablas necesarias de las diferentes partes para crear los diferentes DataFrames

DataFrame1=pd.merge(Data1_FaseIni,Data_FaseNoIni,on='CODI_EXPEDIENT',how='inner')

DataFrame2=pd.merge(Data2_FaseIni,Data_FaseNoIni,on='CODI_EXPEDIENT',how='inner')

DataFrame3=pd.merge(Data3_FaseIni,Data_FaseNoIni,on='CODI_EXPEDIENT',how='inner')

DataFrame3.set_index('CODI_EXPEDIENT', inplace=True)


#Hacemos una descripción final de cada fichero para ver que no hay datos raros

Descripcion_DF1=DataFrame1.describe(include='all')

writer = ExcelWriter('C:/Users/34630/Desktop/TFG ASP/DescripcioDF1.xlsx')

Descripcion_DF1.to_excel(writer, 'Hoja de datos')

writer.save()

Descripcion_DF2=DataFrame2.describe(include='all')

writer = ExcelWriter('C:/Users/34630/Desktop/TFG ASP/DescripcioDF2.xlsx')

Descripcion_DF2.to_excel(writer, 'Hoja de datos')

writer.save()

Descripcion_DF3=DataFrame3.describe(include='all')

writer = ExcelWriter('C:/Users/34630/Desktop/TFG ASP/DescripcioDF3.xlsx')

Descripcion_DF3.to_excel(writer, 'Hoja de datos')

writer.save()
```

```
#Miro los alumnos por cuatri de la fase inicial para ver si las muestras son uniformes
registros=DF5.groupby(['CURS','QUAD'])['CODI_EXPEDIENT'].count()
writer = ExcelWriter('C:/Users/34630/Desktop/TFG ASP/Registros.xlsx')
registros.to_excel(writer, 'Hoja de datos')
writer.save()
#Guardo los DataFrames con to_pickle
DataFrame1.to_pickle('DataFrame1')
DataFrame2.to_pickle('DataFrame2')
DataFrame3.to_pickle('DataFrame3')
writer = ExcelWriter('C:/Users/34630/Desktop/DFS/DF1.xlsx')
DataFrame1.to_excel(writer, 'Hoja de datos')
writer.save()
writer = ExcelWriter('C:/Users/34630/Desktop/DFS/DF2.xlsx')
DataFrame2.to_excel(writer, 'Hoja de datos')
writer.save()
writer = ExcelWriter('C:/Users/34630/Desktop/DFS/DF3.xlsx')
DataFrame3.to_excel(writer, 'Hoja de datos')
writer.save()
print("--- %s seconds ---" % (time.time() - start_time))
```

A1.2. Modelaje y validación

A1.2.1 Regresión Lineal

```
import pandas as pd
import numpy as np
import statistics as st
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

```
from sklearn.metrics import mean_absolute_error

from sklearn.metrics import confusion_matrix

from sklearn.preprocessing import StandardScaler

Ass=['ALG','CALC1','MECFON','QUIM1','FONINFO','GEO','CALC2','TERMO','QUIM2','EXPRE']

Mit=['M-ALG','M-CALC1','M-MECFON','M-QUIM1','M-FONINFO','M-GEO','M-CALC2','M-TERMO','M-QUIM2','M-EXPRE']

Con=['C-ALG','C-CALC1','C-MECFON','C-QUIM1','C-FONINFO','C-GEO','C-CALC2','C-TERMO','C-QUIM2','C-EXPRE']

Sel=['NOTA_ACCES']

Q3 = ['INFO','MEC','EDOS','MATERIALS','ELECTRO','MÈTODES']

df_scaled = StandardScaler()

df1 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame1')

df2 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame2')

df3 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame3')

df=['df1','df2','df3']

#Creo fichero de resultados

f='LR_Results.csv'

MFR=open(f,'w')

MFR.write("Asignatura;DataFrame;R2_OF;RMSE_TRAIN;R2;R2-
ADJ;MSE;RMSE;MAE;RMSE_0;MAE_0;MT_0;MP_0;RMSE_1;MAE_1;MT_1;MP_1;RMSE_2;MAE_2;MT_2;MP_2;RMSE_3;MAE_3;MT_3;
MP_3;N\n")

#Calculo funciones para las matrices de confusión

def calculamatriu(lista,suma):

    M=np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])

    for e in lista:

        M=M+e

    return (M/suma)

def matriu(df,e,CM):

    fff='LINREG - CM de '+e+' para el'+df+'.csv'

    abrir=open(fff,'w')

    for t in CM:

        for p in range(len(t)):

            if p == len(t)-1:

                abrir.write(str(int(round(t[p])))+'\n')

            else:

                abrir.write(str(int(round(t[p])))+';')
```

```
    abrir.close()

#Construyo los modelos para cada modelo y asignatura

for e in df:

    for s in Q3:

        X=[]

        y=[]

        if e=='df1':

            df=df1[Ass+Mit+[s]]

            df=df.dropna().copy()

            X = df[Ass+Mit]

            y=df[s]

        if e=='df2':

            df=df2[Ass+Mit+Con+[s]]

            df=df.dropna().copy()

            X = df[Ass+Mit+Con]

            y=df[s]

        if e=='df3':

            df=df3[Ass+Mit+Con+Sel+[s]]

            df=df.dropna().copy()

            X = df[Ass+Mit+Con+Sel]

            y=df[s]

        MFR.write(s+';'+e+          ';')

        X_scaled = pd.DataFrame(df_scaled.fit_transform(X), columns = X.columns)

        random_seed=[0,11,15]

        R2_OF=[]; RMSE_TRAIN=[]; R2=[]; MSE=[]; RMSE=[]; MAE=[]; RMSE_0=[]; MAE_0=[]; MT_0=[]; MP_0=[]; RMSE_1=[]; MAE_1=[];
MT_1=[]; MP_1=[]; RMSE_2=[]; MAE_2=[]; MT_2=[]; MP_2=[]; RMSE_3=[]; MAE_3=[]; MT_3=[]; MP_3=[]; CM=[]

        suma=0

        for random in random_seed:

            suma=suma+1

            X_train,        X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.33, random_state=random)

            modelo= LinearRegression()

            modelo.fit(X_train,y_train)

            y_pred=modelo.predict(X_test)
```

```
y_pru=modelo.predict(X_train)
R2_OF.append(r2_score(y_train,y_pru))
RMSE_TRAIN.append(mean_squared_error(y_train, y_pru,squared=False))
R2.append(r2_score(y_test,y_pred))
MSE.append(mean_squared_error(y_test, y_pred))
RMSE.append(mean_squared_error(y_test, y_pred,squared=False))
MAE.append(mean_absolute_error(y_test, y_pred))
ll1=[]
ll2=[]
t0_test=[]
t0_pred=[]
t1_test=[]
t1_pred=[]
t2_test=[]
t2_pred=[]
t3_test=[]
t3_pred=[]
for v in y_pred:
    if v<=4:
        ll1.append(0)
    elif 4<v<=6:
        ll1.append(1)
    elif 6<v<=8:
        ll1.append(2)
    elif 8<v:
        ll1.append(3)

for w in range(len(y_test.array)):
    if y_test.array[w]<=4:
        ll2.append(0)
        t0_test.append(y_test.array[w])
        t0_pred.append(y_pred[w])
```

```
            elif 4<y_test.array[w]<=6:
                ll2.append(1)
                t1_test.append(y_test.array[w])
                t1_pred.append(y_pred[w])
            elif 6<y_test.array[w]<=8:
                ll2.append(2)
                t2_test.append(y_test.array[w])
                t2_pred.append(y_pred[w])
            elif 8<y_test.array[w]:
                ll2.append(3)
                t3_test.append(y_test.array[w])
                t3_pred.append(y_pred[w])
        #Calculo matriz de confusión y métricas
        CM.append(confusion_matrix(ll2,ll1))
        RMSE_0.append(mean_squared_error(t0_test,t0_pred,squared=False))
        RMSE_1.append(mean_squared_error(t1_test,t1_pred,squared=False))
        RMSE_2.append(mean_squared_error(t2_test,t2_pred,squared=False))
        RMSE_3.append(mean_squared_error(t3_test,t3_pred,squared=False))
        MAE_0.append(mean_absolute_error(t0_test,t0_pred))
        MAE_1.append(mean_absolute_error(t1_test,t1_pred))
        MAE_2.append(mean_absolute_error(t2_test,t2_pred))
        MAE_3.append(mean_absolute_error(t3_test,t3_pred))
        MT_0.append(st.mean(t0_test))
        MP_0.append(st.mean(t0_pred))
        MT_1.append(st.mean(t1_test))
        MP_1.append(st.mean(t1_pred))
        MT_2.append(st.mean(t2_test))
        MP_2.append(st.mean(t2_pred))
        MT_3.append(st.mean(t3_test))
        MP_3.append(st.mean(t3_pred))
    CMX=calculamatriu(CM,suma)
    medR2=st.mean(R2)
```

```
        r2adj=1-(((1-medR2)*(X_test.shape[0]-1))/(X_test.shape[0]-X_test.shape[1]-1))

        #Escribo resultados finales

        MFR.write(str(round(st.mean(R2_OF),4))+';')

        MFR.write(str(round(st.mean(RMSE_TRAIN),4))+';')

        MFR.write(str(round(st.mean(R2),4))+';')

        MFR.write(str(round(r2adj,4))+';')

        MFR.write(str(round(st.mean(MSE),4))+';')

        MFR.write(str(round(st.mean(RMSE),4))+';')

        MFR.write(str(round(st.mean(MAE),4))+';')

        MFR.write(str(round(st.mean(RMSE_0),4))+';'+str(round(st.mean(MAE_0),4))+';'+
str(round(st.mean(MT_0),4))+';'+str(round(st.mean(MP_0),4))+';')

        MFR.write(str(round(st.mean(RMSE_1),4))+';'+str(round(st.mean(MAE_1),4))+';'+
str(round(st.mean(MT_1),4))+';'+str(round(st.mean(MP_1),4))+';')

        MFR.write(str(round(st.mean(RMSE_2),4))+';'+str(round(st.mean(MAE_2),4))+';'+
str(round(st.mean(MT_2),4))+';'+str(round(st.mean(MP_2),4))+';')

        MFR.write(str(round(st.mean(RMSE_3),4))+';'+str(round(st.mean(MAE_3),4))+';'+
str(round(st.mean(MT_3),4))+';'+str(round(st.mean(MP_3),4))+';')

        MFR.write(str(X_test.shape[0])+'\n')

        A=matriu(e,s,CMX)

#Cambio puntos por comas en el fichero

MFR.close()

dfini=open('LR_Results.csv','r')

a=dfini.read().replace('.',',')

dfini.close()

dffinal=open('LR_Results.csv','w')

dffinal.write(a)

dffinal.close()
```

A1.2.2 Regresión Ridge

```
import pandas as pd

import numpy as np

import statistics as st

from sklearn.linear_model import Ridge

from sklearn.metrics import r2_score

from sklearn.model_selection import train_test_split
```

```
from sklearn.model_selection import GridSearchCV

from sklearn.metrics import r2_score

from sklearn.metrics import mean_squared_error

from sklearn.metrics import mean_absolute_error

from sklearn.metrics import confusion_matrix

from sklearn.preprocessing import StandardScaler

Ass=['ALG','CALC1','MECFON','QUIM1','FONINFO','GEO','CALC2','TERMO','QUIM2','EXPRE']

Mit=['M-ALG','M-CALC1','M-MECFON','M-QUIM1','M-FONINFO','M-GEO','M-CALC2','M-TERMO','M-QUIM2','M-EXPRE']

Con=['C-ALG','C-CALC1','C-MECFON','C-QUIM1','C-FONINFO','C-GEO','C-CALC2','C-TERMO','C-QUIM2','C-EXPRE']

Sel=['NOTA_ACCES']

Q3 = ['INFO','MEC','EDOS','MATERIALS','ELECTRO','MÈTODES']

df_scaled = StandardScaler()

df1 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame1')

df2 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame2')

df3 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame3')

df=['df1','df2','df3']

#Creo fichero de resultados

f='RDG_Results.csv'

MFR=open(f,'w')

MFR.write("Asignatura;DataFrame;R2_OF;RMSE_TRAIN;R2;R2-
ADJ;MSE;RMSE;MAE;RMSE_0;MAE_0;MT_0;MP_0;RMSE_1;MAE_1;MT_1;MP_1;RMSE_2;MAE_2;MT_2;MP_2;RMSE_3;MAE_3;MT_3;
MP_3;N;alpha;best_score\n")

#Calculo funciones para las matrices de confusión

def calculamatriu(lista,suma):

    M=np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])

    for e in lista:

        M=M+e

    return (M/suma)

def matriu(df,e,CM):

    fff='RDG - CM de '+e+' para el'+df+'.csv'

    abrir=open(fff,'w')

    for t in CM:

        for p in range(len(t)):

            if p == len(t)-1:
```

```
            abrir.write(str(int(round(t[p])))+'\n')
        else:
            abrir.write(str(int(round(t[p])))+';')
    abrir.close()
#Selección de hiperparámetro, cosntrucción del modelo y obtención de resultados para cada asignatura y DataFrame
for e in df:
    for s in Q3:
        X=[]
        y=[]
        if e=='df1':
            df=df1[Ass+Mit+[s]]
            df=df.dropna().copy()
            X = df[Ass+Mit]
            y=df[s]
        if e=='df2':
            df=df2[Ass+Mit+Con+[s]]
            df=df.dropna().copy()
            X = df[Ass+Mit+Con]
            y=df[s]
        if e=='df3':
            df=df3[Ass+Mit+Con+Sel+[s]]
            df=df.dropna().copy()
            X = df[Ass+Mit+Con+Sel]
            y=df[s]
        MFR.write(s+';'+e+           ';')
        X_scaled = pd.DataFrame(df_scaled.fit_transform(X), columns = X.columns)
        random_seed=[0,11,15]
        R2_OF=[]; RMSE_TRAIN=[]; R2=[]; MSE=[];RMSE=[]; MAE=[]; RMSE_0=[]; MAE_0=[]; MT_0=[]; MP_0=[]; RMSE_1=[]; MAE_1=[];
MT_1=[]; MP_1=[]; RMSE_2=[]; MAE_2=[]; MT_2=[]; MP_2=[]; RMSE_3=[]; MAE_3=[]; MT_3=[]; MP_3=[]; CM=[]; alpha=[]; best_score=[]
        suma=0
        for random in random_seed:
            suma=suma+1
            X_train,        X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.33, random_state=random )
```

```
parameters = {'alpha':np.arange(0,1,0.01)}
clf = GridSearchCV(Ridge(), parameters,scoring='neg_root_mean_squared_error',cv=3)
clf.fit(X_train,y_train)
parametros=clf.best_params_.values()
alp=0
for q in parametros:
    alp=q
alpha.append(alp)
best_score.append(clf.best_score_)
#Coinstruyo modelo
modelo=Ridge(alpha=alp)
modelo.fit(X_train,y_train)
y_pred=modelo.predict(X_test)
y_pru=modelo.predict(X_train)
#Calculo métricas
R2_OF.append(r2_score(y_train,y_pru))
RMSE_TRAIN.append(mean_squared_error(y_train, y_pru,squared=False))
R2.append(r2_score(y_test,y_pred))
MSE.append(mean_squared_error(y_test, y_pred))
RMSE.append(mean_squared_error(y_test, y_pred,squared=False))
MAE.append(mean_absolute_error(y_test, y_pred))
ll1=[]
ll2=[]
t0_test=[]
t0_pred=[]
t1_test=[]
t1_pred=[]
t2_test=[]
t2_pred=[]
t3_test=[]
t3_pred=[]
for v in y_pred:
```

```
    if v<=4:
        ll1.append(0)
    elif 4<v<=6:
        ll1.append(1)
    elif 6<v<=8:
        ll1.append(2)
    elif 8<v:
        ll1.append(3)


for w in range(len(y_test.array)):
    if y_test.array[w]<=4:
        ll2.append(0)
        t0_test.append(y_test.array[w])
        t0_pred.append(y_pred[w])
    elif 4<y_test.array[w]<=6:
        ll2.append(1)
        t1_test.append(y_test.array[w])
        t1_pred.append(y_pred[w])
    elif 6<y_test.array[w]<=8:
        ll2.append(2)
        t2_test.append(y_test.array[w])
        t2_pred.append(y_pred[w])
    elif 8<y_test.array[w]:
        ll2.append(3)
        t3_test.append(y_test.array[w])
        t3_pred.append(y_pred[w])
CM.append(confusion_matrix(ll2,ll1))
RMSE_0.append(mean_squared_error(t0_test,t0_pred,squared=False))
RMSE_1.append(mean_squared_error(t1_test,t1_pred,squared=False))
RMSE_2.append(mean_squared_error(t2_test,t2_pred,squared=False))
RMSE_3.append(mean_squared_error(t3_test,t3_pred,squared=False))
MAE_0.append(mean_absolute_error(t0_test,t0_pred))
```

```
            MAE_1.append(mean_absolute_error(t1_test,t1_pred))

            MAE_2.append(mean_absolute_error(t2_test,t2_pred))

            MAE_3.append(mean_absolute_error(t3_test,t3_pred))

            MT_0.append(st.mean(t0_test))

            MP_0.append(st.mean(t0_pred))

            MT_1.append(st.mean(t1_test))

            MP_1.append(st.mean(t1_pred))

            MT_2.append(st.mean(t2_test))

            MP_2.append(st.mean(t2_pred))

            MT_3.append(st.mean(t3_test))

            MP_3.append(st.mean(t3_pred))

        CMX=calculamatriu(CM,suma)

        medR2=st.mean(R2)

        r2adj=1-(((1-medR2)*(X_test.shape[0]-1))/(X_test.shape[0]-X_test.shape[1]-1))

        #Escribo resultados

        MFR.write(str(round(st.mean(R2_OF),4))+';')

        MFR.write(str(round(st.mean(RMSE_TRAIN),4))+';')

        MFR.write(str(round(st.mean(R2),4))+';')

        MFR.write(str(round(r2adj,4))+';')

        MFR.write(str(round(st.mean(MSE),4))+';')

        MFR.write(str(round(st.mean(RMSE),4))+';')

        MFR.write(str(round(st.mean(MAE),4))+';')

        MFR.write(str(round(st.mean(RMSE_0),4))+';'+str(round(st.mean(MAE_0),4))+';'+
str(round(st.mean(MT_0),4))+';'+str(round(st.mean(MP_0),4))+';')

        MFR.write(str(round(st.mean(RMSE_1),4))+';'+str(round(st.mean(MAE_1),4))+';'+
str(round(st.mean(MT_1),4))+';'+str(round(st.mean(MP_1),4))+';')

        MFR.write(str(round(st.mean(RMSE_2),4))+';'+str(round(st.mean(MAE_2),4))+';'+
str(round(st.mean(MT_2),4))+';'+str(round(st.mean(MP_2),4))+';')

        MFR.write(str(round(st.mean(RMSE_3),4))+';'+str(round(st.mean(MAE_3),4))+';'+
str(round(st.mean(MT_3),4))+';'+str(round(st.mean(MP_3),4))+';')

        MFR.write(str(X_test.shape[0])+';')

        MFR.write(str(round(int(round(st.mean(alpha)))))+';')

        MFR.write(str(round(st.mean(best_score),4))+'\n')

        A=matriu(e,s,CMX)

MFR.close()
```

```
dfini=open('RDG_Results.csv','r')
a=dfini.read().replace('.',',')
dfini.close()
dffinal=open('RDG_Results.csv','w')
dffinal.write(a)
dffinal.close()
```

A1.2.3 K-Nearest Neighbors

```
import pandas as pd
import numpy as np
import statistics as st
import math
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
Ass=['ALG','CALC1','MECFON','QUIM1','FONINFO','GEO','CALC2','TERMO','QUIM2','EXPRE']
Mit=['M-ALG','M-CALC1','M-MECFON','M-QUIM1','M-FONINFO','M-GEO','M-CALC2','M-TERMO','M-QUIM2','M-EXPRE']
Con=['C-ALG','C-CALC1','C-MECFON','C-QUIM1','C-FONINFO','C-GEO','C-CALC2','C-TERMO','C-QUIM2','C-EXPRE']
Sel=['NOTA_ACCES']
Q3 = ['INFO','MEC','EDOS','MATERIALS','ELECTRO','MÈTODES']
df_scaled = StandardScaler()
df1 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame1')
df2 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame2')
df3 = pd.read_pickle('C:/Users/34630/Desktop/TFG ASP/DataFrame3')
df=['df1','df2','df3']
#Creo fichero de resultados
f='KNNR_Results2.csv'
```

```
MFR=open(f,'w')

MFR.write("Asignatura;DataFrame;R2_OF;RMSE_TRAIN;R2;R2-
ADJ;MSE;RMSE;MAE;RMSE_0;MAE_0;MT_0;MP_0;RMSE_1;MAE_1;MT_1;MP_1;RMSE_2;MAE_2;MT_2;MP_2;RMSE_3;MAE_3;MT_3;
MP_3;N;K;best_score\n")

#Calculo funciones para las matrices de confusión

def calculamatriu(lista,suma):

    M=np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])

    for e in lista:

        M=M+e

    return (M/suma)

def matriu(df,e,CM):

    fff='KNNR - CM de '+e+' para el'+df+'.csv'

    abrir=open(fff,'w')

    for t in CM:

        for p in range(len(t)):

            if p == len(t)-1:

                abrir.write(str(int(round(t[p])))+'\n')

            else:

                abrir.write(str(int(round(t[p])))+';')

    abrir.close()

#Selección de hiperparámetro, cosntrucción del modelo y obtención de resultados para cada asignatura y DataFrame

for e in df:

    for s in Q3:

        X=[]

        y=[]

        if e=='df1':

            df=df1[Ass+Mit+[s]]

            df=df.dropna().copy()

            X = df[Ass+Mit]

            y=df[s]

        if e=='df2':

            df=df2[Ass+Mit+Con+[s]]

            df=df.dropna().copy()

            X = df[Ass+Mit+Con]
```

```
        y=df[s]
    if e=='df3':
        df=df3[Ass+Mit+Con+Sel+[s]]
        df=df.dropna().copy()
        X = df[Ass+Mit+Con+Sel]
        y=df[s]
    MFR.write(s+';'+e+          ';')
    X_scaled = pd.DataFrame(df_scaled.fit_transform(X), columns = X.columns)
    random_seed=[0,11,15]
    R2_OF=[]; RMSE_TRAIN=[]; R2=[]; MSE=[];RMSE=[]; MAE=[]; RMSE_0=[]; MAE_0=[]; MT_0=[]; MP_0=[]; RMSE_1=[]; MAE_1=[];
MT_1=[]; MP_1=[]; RMSE_2=[]; MAE_2=[]; MT_2=[]; MP_2=[]; RMSE_3=[]; MAE_3=[]; MT_3=[]; MP_3=[]; CM=[]; K=[]; best_score=[]
    suma=0
    for random in random_seed:
        suma=suma+1
        X_train,        X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.33, random_state=random)
        klim=2*math.sqrt(len(X_train))
        parameters = {'n_neighbors':np.arange(1,int(klim),1)}
        clf = GridSearchCV(KNeighborsRegressor(), parameters,scoring='neg_root_mean_squared_error',cv=3)
        clf.fit(X_train,y_train)
        parametros=clf.best_params_.values()
        k=0
        for q in parametros:
            k=q
        K.append(k)
        best_score.append(clf.best_score_)
        #Construyo modelo
        modelo=KNeighborsRegressor(n_neighbors=k)
        modelo.fit(X_train,y_train)
        y_pred=modelo.predict(X_test)
        y_pru=modelo.predict(X_train)
        #Calculo métricas
        R2_OF.append(r2_score(y_train,y_pru))
        RMSE_TRAIN.append(mean_squared_error(y_train, y_pru,squared=False))
```

```
R2.append(r2_score(y_test,y_pred))
MSE.append(mean_squared_error(y_test, y_pred))
RMSE.append(mean_squared_error(y_test, y_pred,squared=False))
MAE.append(mean_absolute_error(y_test, y_pred))
ll1=[]
ll2=[]
t0_test=[]
t0_pred=[]
t1_test=[]
t1_pred=[]
t2_test=[]
t2_pred=[]
t3_test=[]
t3_pred=[]
for v in y_pred:
    if v<=4:
        ll1.append(0)
    elif 4<v<=6:
        ll1.append(1)
    elif 6<v<=8:
        ll1.append(2)
    elif 8<v:
        ll1.append(3)

for w in range(len(y_test.array)):
    if y_test.array[w]<=4:
        ll2.append(0)
        t0_test.append(y_test.array[w])
        t0_pred.append(y_pred[w])
    elif 4<y_test.array[w]<=6:
        ll2.append(1)
        t1_test.append(y_test.array[w])
```

```
            t1_pred.append(y_pred[w])
        elif 6<y_test.array[w]<=8:
            ll2.append(2)
            t2_test.append(y_test.array[w])
            t2_pred.append(y_pred[w])
        elif 8<y_test.array[w]:
            ll2.append(3)
            t3_test.append(y_test.array[w])
            t3_pred.append(y_pred[w])
    CM.append(confusion_matrix(ll2,ll1))
    RMSE_0.append(mean_squared_error(t0_test,t0_pred,squared=False))
    RMSE_1.append(mean_squared_error(t1_test,t1_pred,squared=False))
    RMSE_2.append(mean_squared_error(t2_test,t2_pred,squared=False))
    RMSE_3.append(mean_squared_error(t3_test,t3_pred,squared=False))
    MAE_0.append(mean_absolute_error(t0_test,t0_pred))
    MAE_1.append(mean_absolute_error(t1_test,t1_pred))
    MAE_2.append(mean_absolute_error(t2_test,t2_pred))
    MAE_3.append(mean_absolute_error(t3_test,t3_pred))
    MT_0.append(st.mean(t0_test))
    MP_0.append(st.mean(t0_pred))
    MT_1.append(st.mean(t1_test))
    MP_1.append(st.mean(t1_pred))
    MT_2.append(st.mean(t2_test))
    MP_2.append(st.mean(t2_pred))
    MT_3.append(st.mean(t3_test))
    MP_3.append(st.mean(t3_pred))
CMX=calculamatriu(CM,suma)
medR2=st.mean(R2)
r2adj=1-(((1-medR2)*(X_test.shape[0]-1))/(X_test.shape[0]-X_test.shape[1]-1))
#Escribo resultados
MFR.write(str(round(st.mean(R2_OF),4))+';')
MFR.write(str(round(st.mean(RMSE_TRAIN),4))+';')
```

```
        MFR.write(str(round(st.mean(R2),4))+';')

        MFR.write(str(round(r2adj,4))+';')

        MFR.write(str(round(st.mean(MSE),4))+';')

        MFR.write(str(round(st.mean(RMSE),4))+';')

        MFR.write(str(round(st.mean(MAE),4))+';')

        MFR.write(str(round(st.mean(RMSE_0),4))+';'+str(round(st.mean(MAE_0),4))+';'+
str(round(st.mean(MT_0),4))+';'+str(round(st.mean(MP_0),4))+';')

        MFR.write(str(round(st.mean(RMSE_1),4))+';'+str(round(st.mean(MAE_1),4))+';'+
str(round(st.mean(MT_1),4))+';'+str(round(st.mean(MP_1),4))+';')

        MFR.write(str(round(st.mean(RMSE_2),4))+';'+str(round(st.mean(MAE_2),4))+';'+
str(round(st.mean(MT_2),4))+';'+str(round(st.mean(MP_2),4))+';')

        MFR.write(str(round(st.mean(RMSE_3),4))+';'+str(round(st.mean(MAE_3),4))+';'+
str(round(st.mean(MT_3),4))+';'+str(round(st.mean(MP_3),4))+';')

        MFR.write(str(X_test.shape[0])+';')

        MFR.write(str(round(int(round(st.mean(K)))))+';')

        MFR.write(str(round(st.mean(best_score),4))+'\n')

        A=matriu(e,s,CMX)

MFR.close()

dfini=open('KNNR_Results2.csv','r')

a=dfini.read().replace('.',',')

dfini.close()

dffinal=open('KNNR_Results2.csv','w')

dffinal.write(a)

dffinal.close()
```

Anexo 2

Regresión Lineal

Informática

DataFrame 1	0-4	4-6	6-8	8-10
0-4	23	112	18	1
4-6	10	182	86	4
6-8	5	117	133	17
8-10	0	18	74	41

RMSE	MAE	Media Test	Media Predicciones
3,0747	2,7416	2,2931	5,0222
1,0183	0,8039	5,3042	5,6013
1,3708	1,0972	6,9494	6,2037
2,0388	1,7229	8,9903	7,3467

Tabla 1: Matriz de confusión y métricas por tramos para informática con regresión lineal para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	27	107	19	1
4-6	12	177	90	3
6-8	5	116	135	17
8-10	1	17	76	40

RMSE	MAE	Media Test	Media Predicciones
3,0545	2,7142	2,2931	4,9868
1,038	0,8192	5,3042	5,6135
1,3805	1,0978	6,9494	6,2047
2,0273	1,7154	8,9903	7,3446

Tabla 2: Matriz de confusión y métricas por tramos para informática con regresión lineal para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	23	86	21	0
4-6	21	165	79	3
6-8	5	100	132	15
8-10	1	14	73	31

RMSE	MAE	Media Test	Media Predicciones
2,9383	2,6162	2,3387	4,9234
1,0913	0,8611	5,2911	5,5
1,3698	1,0821	6,9671	6,2174
2,1473	1,8567	9,0542	7,2391

Tabla 3: Matriz de confusión y métricas por tramos para informática con regresión lineal para el DataFrame 3.

EDOS

DataFrame 1	0-4	4-6	6-8	8-10
0-4	10	95	2	0
4-6	8	353	48	0
6-8	2	175	108	5
8-10	0	4	19	16

RMSE	MAE	Media Test	Media Predicciones
2,8829	2,513	2,2441	4,7565
0,7342	0,6012	5,265	5,1963
1,2939	1,0764	6,7996	5,8732
1,5826	1,2449	8,7881	7,5938

Tabla 4: Matriz de confusión y métricas por tramos para EDOS con regresión lineal para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	14	90	3	0
4-6	21	334	54	0
6-8	3	163	120	5
8-10	0	4	21	15

RMSE	MAE	Media Test	Media Predicciones
2,8612	2,4677	2,2441	4,693
0,7893	0,6385	5,265	5,184
1,2506	1,0267	6,7996	5,9119
1,5807	1,2729	8,7881	7,5412

Tabla 5: Matriz de confusión y métricas por tramos para EDOS con regresión lineal para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	11	91	7	0
4-6	20	277	58	0
6-8	2	154	124	4
8-10	0	3	14	10

RMSE	MAE	Media Test	Media Predicciones
2,8938	2,4669	2,3425	4,7967
0,8076	0,6578	5,2625	5,2597
1,2235	0,9983	6,8142	5,9702
1,4965	1,2729	8,7436	7,4749

Tabla 6: Matriz de confusión y métricas por tramos para EDOS con regresión lineal para el DataFrame 3.

Electro

DataFrame 1	0-4	4-6	6-8	8-10
0-4	63	136	5	0
4-6	39	305	39	0
6-8	6	128	75	8
8-10	0	2	15	7

RMSE	MAE	Media Test	Media Predicciones
2,1171	1,7401	2,6367	4,3524
0,8571	0,6957	5,2274	4,9454
1,418	1,1704	6,7538	5,7671
1,4452	1,1843	8,6472	7,5047

Tabla 7: Matriz de confusión y métricas por tramos para electromagnetismo con regresión lineal para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	65	134	5	0
4-6	44	299	41	0
6-8	7	125	77	8
8-10	0	2	15	7

RMSE	MAE	Media Test	Media Predicciones
2,1266	1,75	2,6367	4,3545
0,8662	0,6985	5,2274	4,9535
1,4009	1,1517	6,7538	5,7746
1,4697	1,2136	8,6472	7,4618

Tabla 8: Matriz de confusión y métricas por tramos para electromagnetismo con regresión lineal para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	56	141	4	0
4-6	32	263	34	0
6-8	6	118	81	4
8-10	0	0	17	2

RMSE	MAE	Media Test	Media Predicciones
2,0496	1,7058	2,7345	4,4019
0,8472	0,6751	5,1806	4,9968
1,3786	1,1367	6,7534	5,7745
1,3712	1,2043	8,6114	7,4155

Tabla 9: Matriz de confusión y métricas por tramos para electromagnetismo con regresión lineal para el DataFrame 3.

Materials

DataFrame 1	0-4	4-6	6-8	8-10
0-4	53	152	3	0
4-6	38	355	40	0
6-8	4	128	65	0
8-10	0	1	9	1

RMSE	MAE	Media Test	Media Predicciones
2,1427	1,7864	2,652	4,4168
0,8373	0,6858	5,1893	4,9276
1,4477	1,233	6,7455	5,5871
1,8603	1,6298	8,7198	7,09

Tabla 10: Matriz de confusión y métricas por tramos para materiales con regresión lineal para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	55	150	3	0
4-6	44	349	40	0
6-8	4	128	66	0
8-10	0	1	10	1

RMSE	MAE	Media Test	Media Predicciones
2,1561	1,7974	2,652	4,4206
0,8484	0,6892	5,1893	4,9337
1,4256	1,2122	6,7455	5,6002
1,8907	1,6802	8,7198	7,0396

Tabla 11: Matriz de confusión y métricas por tramos para materiales con regresión lineal para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	58	132	2	0
4-6	53	318	30	0
6-8	4	115	59	0
8-10	0	1	7	1

RMSE	MAE	Media Test	Media Predicciones
2,0513	1,6898	2,6734	4,3357
0,8697	0,7075	5,1941	4,8602
1,4283	1,2106	6,7186	5,5906
1,9112	1,7269	8,6671	6,9403

Tabla 12: Matriz de confusión y métricas por tramos para materiales con regresión lineal para el DataFrame 3.

Mecánica

DataFrame 1	0-4	4-6	6-8	8-10
0-4	276	92	1	0
4-6	144	178	23	0
6-8	12	57	31	4
8-10	0	7	8	1

RMSE	MAE	Media Test	Media Predicciones
1,8037	1,3962	2,3208	3,5503
1,2691	1,0696	5,114	4,3486
1,7045	1,3895	6,7781	5,5105
2,9753	2,6739	9,0916	6,4243

Tabla 13: Matriz de confusión y métricas por tramos para mecánica con regresión lineal para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	268	99	1	0
4-6	130	192	22	0
6-8	13	57	32	2
8-10	0	7	8	1

RMSE	MAE	Media Test	Media Predicciones
1,8074	1,4042	2,3208	3,5229
1,2573	1,0411	5,114	4,3723
1,712	1,3945	6,7781	5,4815
2,981	2,7126	9,0916	6,379

Tabla 14: Matriz de confusión y métricas por tramos para mecánica con regresión lineal para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	254	91	1	0
4-6	129	154	19	0
6-8	12	59	26	1
8-10	1	6	6	2

RMSE	MAE	Media Test	Media Predicciones
1,7199	1,3546	2,3959	3,5186
1,3355	1,1107	5,1127	4,278
1,8031	1,4935	6,7334	5,3184
3,0203	2,674	9,1554	6,4826

Tabla 15: Matriz de confusión y métricas por tramos para mecánica con regresión lineal para el DataFrame 3.

Métodos

DataFrame 1	0-4	4-6	6-8	8-10
0-4	0	71	18	0
4-6	2	226	89	0
6-8	0	167	190	6
8-10	0	13	51	11

RMSE	MAE	Media Test	Media Predicciones
3,3759	3,0384	2,3189	5,3569
0,8448	0,675	5,3241	5,6823
1,1392	0,9418	6,9149	6,1988
1,9674	1,7406	8,7337	7,0005

Tabla 16: Matriz de confusión y métricas por tramos para métodos con regresión lineal para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	3	68	19	0
4-6	2	221	94	0
6-8	0	162	196	6
8-10	0	12	53	10

RMSE	MAE	Media Test	Media Predicciones
3,3706	3,0144	2,3189	5,3333
0,8558	0,6867	5,3241	5,6861
1,1447	0,9376	6,9149	6,2008
1,9594	1,7388	8,7337	7,0004

Tabla 17: Matriz de confusión y métricas por tramos para métodos con regresión lineal para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	3	56	15	0
4-6	4	184	115	1
6-8	0	133	204	6
8-10	0	6	43	7

RMSE	MAE	Media Test	Media Predicciones
3,2155	2,8522	2,4919	5,3355
0,9427	0,7606	5,3155	5,7776
1,0884	0,8728	6,9157	6,2653
1,8437	1,6355	8,6934	7,0621

Tabla 18: Matriz de confusión y métricas por tramos para métodos con regresión lineal para el DataFrame 3.

Regresión Ridge

Informática

DataFrame 1	0-4	4-6	6-8	8-10
0-4	14	123	16	1
4-6	6	188	83	3
6-8	3	123	131	16
8-10	0	19	74	41

RMSE	MAE	Media Test	Media Predicciones
3,0895	2,768	2,2931	5,0556
0,9766	0,7665	5,3042	5,604
1,3528	1,0913	6,9494	6,196
2,0546	1,741	8,9903	7,3289

Tabla 19: Matriz de confusión y métricas por tramos para informática con regresión ridge para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	23	112	18	1
4-6	8	184	87	2
6-8	5	116	137	15
8-10	0	18	78	37

RMSE	MAE	Media Test	Media Predicciones
3,0703	2,7403	2,2931	5,0226
0,9919	0,7834	5,3042	5,6225
1,3507	1,0781	6,9494	6,1991
2,0529	1,7485	8,9903	7,2956

Tabla 20: Matriz de confusión y métricas por tramos para informática con regresión ridge para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	20	90	19	0
4-6	15	174	78	2
6-8	4	103	135	11
8-10	1	15	74	29

RMSE	MAE	Media Test	Media Predicciones
2,9532	2,6482	2,3387	4,9671
1,0175	0,8091	5,2911	5,5142
1,3255	1,0543	6,9671	6,2088
2,1915	1,9136	9,0542	7,1727

Tabla 21: Matriz de confusión y métricas por tramos para informática con regresión ridge para el DataFrame 3.

EDOS

DataFrame 1	0-4	4-6	6-8	8-10
0-4	5	100	2	0
4-6	6	357	47	0
6-8	0	179	106	5
8-10	0	4	20	16

RMSE	MAE	Media Test	Media Predicciones
2,9105	2,5479	2,2441	4,792
0,7041	0,5784	5,265	5,195
1,2854	1,0779	6,7996	5,8602
1,6115	1,2791	8,7881	7,5482

Tabla 22: Matriz de confusión y métricas por tramos para EDOS con regresión ridge para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	11	93	3	0
4-6	17	340	52	0
6-8	2	169	114	5
8-10	0	4	22	14

RMSE	MAE	Media Test	Media Predicciones
2,8737	2,4981	2,2441	4,7335
0,7645	0,6194	5,265	5,1899
1,2568	1,0388	6,7996	5,894
1,6038	1,2961	8,7881	7,5124

Tabla 23: Matriz de confusión y métricas por tramos para EDOS con regresión ridge para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	9	95	5	0
4-6	13	287	55	0
6-8	1	158	121	4
8-10	0	3	15	8

RMSE	MAE	Media Test	Media Predicciones
2,8934	2,4934	2,3425	4,8317
0,7696	0,6274	5,2625	5,2718
1,227	1,0083	6,8142	5,9487
1,5508	1,3268	8,7436	7,421

Tabla 24: Matriz de confusión y métricas por tramos para EDOS con regresión ridge para el DataFrame 3.

Electro

DataFrame 1	0-4	4-6	6-8	8-10
0-4	55	144	5	0
4-6	32	313	38	0
6-8	5	131	74	7
8-10	0	2	15	7

RMSE	MAE	Media Test	Media Predicciones
2,1386	1,7702	2,6367	4,393
0,8324	0,6789	5,2274	4,9406
1,4093	1,1719	6,7538	5,7507
1,4641	1,2103	8,6472	7,4725

Tabla 25: Matriz de confusión y métricas por tramos para electromagnetismo con regresión ridge para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	56	143	5	0
4-6	34	312	39	0
6-8	6	130	76	5
8-10	0	2	17	5

RMSE	MAE	Media Test	Media Predicciones
2,1439	1,7738	2,6367	4,3879
0,8332	0,6706	5,2274	4,9601
1,3896	1,1481	6,7538	5,7424
1,5289	1,3017	8,6472	7,3518

Tabla 26: Matriz de confusión y métricas por tramos para electromagnetismo con regresión ridge para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	51	146	4	0
4-6	23	273	33	0
6-8	5	119	81	3
8-10	0	0	16	2

RMSE	MAE	Media Test	Media Predicciones
2,0541	1,7181	2,7345	4,4265
0,8081	0,6464	5,1806	5,0017
1,3627	1,1324	6,7534	5,7523
1,441	1,295	8,6114	7,3179

Tabla 27: Matriz de confusión y métricas por tramos para electromagnetismo con regresión ridge para el DataFrame 3.

Materiales

DataFrame 1	0-4	4-6	6-8	8-10
0-4	45	159	3	0
4-6	33	364	36	0
6-8	3	132	62	0
8-10	0	1	9	2

RMSE	MAE	Media Test	Media Predicciones
2,1473	1,7941	2,652	4,4313
0,8129	0,668	5,1893	4,9233
1,4535	1,2468	6,7455	5,5715
1,8594	1,6271	8,7198	7,0927

Tabla 28: Matriz de confusión y métricas por tramos para materiales con regresión ridge para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	51	153	3	0
4-6	39	358	36	0
6-8	3	133	61	0
8-10	0	1	10	0

RMSE	MAE	Media Test	Media Predicciones
2,1552	1,7999	2,652	4,4284
0,8069	0,6581	5,1893	4,9347
1,4305	1,2333	6,7455	5,5657
1,9376	1,7529	8,7198	6,9669

Tabla 29: Matriz de confusión y métricas por tramos para materiales con regresión ridge para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	54	137	1	0
4-6	43	332	26	0
6-8	4	122	53	0
8-10	0	1	8	0

RMSE	MAE	Media Test	Media Predicciones
2,059	1,712	2,6734	4,3628
0,8225	0,6717	5,1941	4,8686
1,4376	1,2412	6,7186	5,5413
1,9804	1,8302	8,6671	6,8369

Tabla 30: Matriz de confusión y métricas por tramos para materiales con regresión ridge para el DataFrame 3.

Mecánica

DataFrame 1	0-4	4-6	6-8	8-10
0-4	281	87	1	0
4-6	145	180	19	0
6-8	12	57	32	2
8-10	1	6	8	1

RMSE	MAE	Media Test	Media Predicciones
1,8112	1,401	2,3208	3,5821
1,2452	1,0583	5,114	4,334
1,7129	1,4119	6,7781	5,4696
3,0179	2,7309	9,0916	6,3657

Tabla 31: Matriz de confusión y métricas por tramos para mecánica con regresión ridge para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	274	93	1	0
4-6	133	196	16	0
6-8	13	58	32	1
8-10	0	7	8	1

RMSE	MAE	Media Test	Media Predicciones
1,808	1,4059	2,3208	3,57
1,2212	1,0251	5,114	4,3535
1,7188	1,4241	6,7781	5,4253
3,0668	2,8182	9,0916	6,2734

Tabla 32: Matriz de confusión y métricas por tramos para mecánica con regresión ridge para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	257	89	1	0
4-6	132	154	16	0
6-8	13	60	24	1
8-10	1	6	7	1

RMSE	MAE	Media Test	Media Predicciones
1,7017	1,3383	2,3959	3,5505
1,3149	1,1042	5,1127	4,2518
1,8111	1,5164	6,7334	5,2788
3,0777	2,7543	9,1554	6,4011

Tabla 33: Matriz de confusión y métricas por tramos para mecánica con regresión ridge para el DataFrame 3.

Métodos

DataFrame 1	0-4	4-6	6-8	8-10
0-4	0	74	15	0
4-6	0	233	84	0
6-8	0	174	184	6
8-10	0	13	51	11

RMSE	MAE	Media Test	Media Predicciones
3,3879	3,0596	2,3189	5,3785
0,8085	0,6392	5,3241	5,6861
1,1315	0,943	6,9149	6,1884
1,975	1,7508	8,7337	6,9917

Tabla 34: Matriz de confusión y métricas por tramos para métodos con regresión ridge para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	1	70	18	0
4-6	0	225	92	0
6-8	0	166	194	5
8-10	0	13	55	8

RMSE	MAE	Media Test	Media Predicciones
3,3843	3,0426	2,3189	5,3609
0,8272	0,6645	5,3241	5,705
1,112	0,9192	6,9149	6,1907
1,9953	1,7933	8,7337	6,9435

Tabla 35: Matriz de confusión y métricas por tramos para métodos con regresión ridge para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	1	57	16	0
4-6	1	186	115	1
6-8	0	140	201	3
8-10	0	7	44	5

RMSE	MAE	Media Test	Media Predicciones
3,2178	2,882	2,4919	5,3673
0,9044	0,7273	5,3155	5,7902
1,0579	0,8588	6,9157	6,2505
1,8859	1,6924	8,6934	7,005

Tabla 36: Matriz de confusión y métricas por tramos para métodos con regresión ridge para el DataFrame 3.

K-Nearest Neighbors

Informática

DataFrame 1	0-4	4-6	6-8	8-10
0-4	7	129	17	0
4-6	2	198	81	1
6-8	3	122	136	11
8-10	0	21	81	32

RMSE	MAE	Media Test	Media Predicciones
3,1665	2,8683	2,2931	5,1592
0,907	0,7092	5,3042	5,6211
1,2939	1,0541	6,9494	6,1582
2,14	1,8752	8,9903	7,1299

Tabla 37: Matriz de confusión y métricas por tramos para informática con K-Nearest Neighbors para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	5	129	19	0
4-6	1	196	84	0
6-8	2	117	144	10
8-10	0	18	83	32

RMSE	MAE	Media Test	Media Predicciones
3,228	2,949	2,2931	5,2416
0,914	0,7167	5,3042	5,7257
1,2277	0,9928	6,9494	6,2251
2,1143	1,8556	8,9903	7,1501

Tabla 38: Matriz de confusión y métricas por tramos para informática con K-Nearest Neighbors para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	9	103	17	0
4-6	3	187	77	0
6-8	1	110	131	10
8-10	1	17	78	23

RMSE	MAE	Media Test	Media Predicciones
3,1382	2,8608	2,3387	5,1977
0,9013	0,6931	5,2911	5,6412
1,2135	0,9859	6,9671	6,2183
2,2835	2,0549	9,0542	7,005

Tabla 39: Matriz de confusión y métricas por tramos para informática con K-Nearest Neighbors para el DataFrame 3.

EDOS

DataFrame 1	0-4	4-6	6-8	8-10
0-4	10	96	1	0
4-6	14	360	35	0
6-8	1	184	103	3
8-10	0	4	25	10

RMSE	MAE	Media Test	Media Predicciones
2,9037	2,529	2,2441	4,7731
0,7023	0,5643	5,265	5,1505
1,3052	1,0997	6,7996	5,7785
1,7796	1,5473	8,7881	7,2409

Tabla 40: Matriz de confusión y métricas por tramos para EDOS con K-Nearest Neighbors para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	2	104	1	0
4-6	0	368	40	0
6-8	0	182	106	2
8-10	0	5	26	9

RMSE	MAE	Media Test	Media Predicciones
3,053	2,7083	2,2441	4,9524
0,6259	0,5034	5,265	5,2698
1,234	1,0506	6,7996	5,8255
1,7735	1,5668	8,7881	7,2213

Tabla 41: Matriz de confusión y métricas por tramos para EDOS con K-Nearest Neighbors para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	1	106	2	0
4-6	2	315	38	0
6-8	0	175	108	0
8-10	0	4	21	2

RMSE	MAE	Media Test	Media Predicciones
3,0147	2,6675	2,3425	5,01
0,6443	0,5196	5,2625	5,3354
1,2058	1,0231	6,8142	5,8673
1,8126	1,6839	8,7436	7,0598

Tabla 42: Matriz de confusión y métricas por tramos para EDOS con K-Nearest Neighbors para el DataFrame 3.

Electro

DataFrame 1	0-4	4-6	6-8	8-10
0-4	47	152	4	0
4-6	33	320	32	0
6-8	5	135	77	1
8-10	0	2	21	1

RMSE	MAE	Media Test	Media Predicciones
2,1893	1,8376	2,6367	4,4632
0,82	0,6553	5,2274	4,9337
1,4082	1,1752	6,7538	5,6718
1,6996	1,5458	8,6472	7,1014

Tabla 43: Matriz de confusión y métricas por tramos para electro con K-Nearest Neighbors para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	31	169	3	0
4-6	22	328	34	0
6-8	2	140	74	1
8-10	0	2	21	1

RMSE	MAE	Media Test	Media Predicciones
2,2436	1,9043	2,6367	4,5378
0,7722	0,6213	5,2274	5,0044
1,3787	1,1525	6,7538	5,6825
1,7387	1,5878	8,6472	7,0594

Tabla 44: Matriz de confusión y métricas por tramos para electro con K-Nearest Neighbors para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	22	176	2	0
4-6	13	291	25	0
6-8	3	138	68	0
8-10	0	1	17	0

RMSE	MAE	Media Test	Media Predicciones
2,1688	1,8682	2,7345	4,5987
0,7244	0,5823	5,1806	5,0636
1,3555	1,1424	6,7534	5,6966
1,6754	1,5729	8,6114	7,0385

Tabla 45: Matriz de confusión y métricas por tramos para electro con K-Nearest Neighbors para el DataFrame 3.

Materiales

DataFrame 1	0-4	4-6	6-8	8-10
0-4	38	167	3	0
4-6	30	371	32	0
6-8	4	134	60	0
8-10	0	1	10	0

RMSE	MAE	Media Test	Media Predicciones
2,2274	1,8774	2,652	4,5251
0,7564	0,6133	5,1893	4,958
1,4484	1,2563	6,7455	5,5353
2,0937	1,9164	8,7198	6,8034

Tabla 46: Matriz de confusión y métricas por tramos para materiales con K-Nearest Neighbors para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	33	172	3	0
4-6	18	384	31	0
6-8	2	140	55	0
8-10	0	1	10	0

RMSE	MAE	Media Test	Media Predicciones
2,2812	1,9403	2,652	4,5876
0,7119	0,5783	5,1893	5,0081
1,428	1,2493	6,7455	5,5333
2,0758	1,9359	8,7198	6,7839

Tabla 47: Matriz de confusión y métricas por tramos para materiales con K-Nearest Neighbors para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	37	153	1	0
4-6	23	353	25	0
6-8	2	132	45	0
8-10	0	1	8	0

RMSE	MAE	Media Test	Media Predicciones
2,1561	1,8286	2,6734	4,4984
0,7497	0,6082	5,1941	4,9306
1,4345	1,256	6,7186	5,5015
2,1757	2,0272	8,6671	6,6399

Tabla 48: Matriz de confusión y métricas por tramos para materiales con K-Nearest Neighbors para el DataFrame 3.

Mecánica

DataFrame 1	0-4	4-6	6-8	8-10
0-4	277	91	1	0
4-6	141	188	14	0
6-8	14	59	31	0
8-10	0	7	10	0

RMSE	MAE	Media Test	Media Predicciones
1,8498	1,4217	2,3208	3,5959
1,2178	1,0255	5,114	4,3223
1,7838	1,5011	6,7781	5,3268
3,2946	3,1322	9,0916	5,9594

Tabla 49: Matriz de confusión y métricas por tramos para mecánica con K-Nearest Neighbors para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	252	116	1	0
4-6	118	211	14	0
6-8	12	59	33	0
8-10	0	7	9	0

RMSE	MAE	Media Test	Media Predicciones
1,8955	1,4826	2,3208	3,6807
1,1758	0,9789	5,114	4,3764
1,7665	1,4751	6,7781	5,3528
3,3035	3,1347	9,0916	5,9569

Tabla 50: Matriz de confusión y métricas por tramos para mecánica con K-Nearest Neighbors para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	239	107	0	0
4-6	119	174	9	0
6-8	13	64	22	0
8-10	1	6	8	0

RMSE	MAE	Media Test	Media Predicciones
1,7854	1,415	2,3959	3,6864
1,2103	1,0046	5,1127	4,2911
1,8312	1,5612	6,7334	5,1846
3,3784	3,2115	9,1554	5,9439

Tabla 51: Matriz de confusión y métricas por tramos para mecánica con K-Nearest Neighbors para el DataFrame 3.

Métodos

DataFrame 1	0-4	4-6	6-8	8-10
0-4	0	74	15	0
4-6	0	218	98	0
6-8	0	162	198	5
8-10	0	13	54	8

RMSE	MAE	Media Test	Media Predicciones
3,4841	3,1611	2,3189	5,4789
0,7959	0,6411	5,3241	5,7695
1,0639	0,8815	6,9149	6,1717
2,077	1,9095	8,7337	6,825

Tabla 52: Matriz de confusión y métricas por tramos para métodos con K-Nearest Neighbors para el DataFrame 1.

DataFrame 2	0-4	4-6	6-8	8-10
0-4	0	75	14	0
4-6	0	218	99	0
6-8	0	167	193	4
8-10	0	14	54	7

RMSE	MAE	Media Test	Media Predicciones
3,4752	3,1741	2,3189	5,493
0,7768	0,6218	5,3241	5,7803
1,0422	0,8675	6,9149	6,1885
2,0564	1,8916	8,7337	6,842

Tabla 53: Matriz de confusión y métricas por tramos para métodos con K-Nearest Neighbors para el DataFrame 2.

DataFrame 3	0-4	4-6	6-8	8-10
0-4	0	57	17	0
4-6	1	189	112	1
6-8	0	134	204	6
8-10	0	5	43	8

RMSE	MAE	Media Test	Media Predicciones
3,3601	3,0617	2,4919	5,5536
0,8532	0,6911	5,3155	5,8589
0,9934	0,8179	6,9157	6,2528
1,8972	1,7175	8,6934	6,9765

Tabla 54: Matriz de confusión y métricas por tramos para métodos con K-Nearest Neighbors para el DataFrame 3.