0. Objetivo del Proyecto

El objetivo del proyecto consiste en el análisis y modelado de un dataset de un conjunto de clientes en el cual se registró si se subscribieron a un depósito a plazo fijo. En base a este análisis y modelado se debe predecir si el cliente se va a subscribir a dicho producto. Para ello El dataset se llama bank-additional-full y corresponden a datos de antiguos clientes de una entiedad bancaria portuguesa obtenidos en la url de la uci: https://archive.ics.uci.edu/ml/datasets/bank+marketing

1. Lectura del dataset

Se descarga el csv de la url descrita en el directorio de trabajo. Mediante la librería pandas se descarga en memoria como dataframe introduciendo el caracter separador ";" para su correcta visualización. Es importante que se realimente con nuevos datos para su actualización y mejora en la predicción.

In [1]:
```
# 1. Se desccarga librería pandas para carga en memoria del dataset.
import pandas as pd
# 2. Directorio donde se encuentra el recurso
url='bank-additional-full.csv'
# 3. Carga en memoria como dataframe del csv
df = pd.read_csv(url,sep=";")
# 4. Visualizacion mediante head  de los 5 primeros registros del datafr
df.head()
```

Out[1]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_v
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	57	services	married	high.school	unknown	no	no	telephone	may	
2	37	services	married	high.school	no	yes	no	telephone	may	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	
4	56	services	married	high.school	no	no	yes	telephone	may	

5 rows × 21 columns

2. Análisis Inicial y visualización.

2.1. Dimension y metadatos asociados al dataset

Un análisis previo consiste en obtener la dimensión y los metadatos del dataset. Se observa que el dataset tiene un tamaño medio de 41188 datos de 20 variables y 1 variable objetivo. Además se visualiza los nombres de las columnas, nulos asociados y tipo de datos que contiene. Así como la memoria ocupada por el dataset: 6.6 MB Como la variable dependiente o target es categórica, nos enfrentamos a un problema de clasificación binaria.

In [15]:
```
# 1. Mediante el método shape se obtiene la dimension del dataset origin
```

```
# 41188 filas y 21 columnas
df.shape
```

Out[15]:
```
(41188, 21)
```

In [16]:
```
# 1. Con la función info() de dataframe se obtiene todos los metadatos a
# nombre de columnas, nulos asociados, tipo datos y memoria ocupada.
# Balance: 10 variables numéricas y 10 categóricas. Con variable dependi
# Memoria ocupada: 6.6 MB.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

2.2. Descripción detallada del dataset

Decripcion detallada de las features predictoras :

a) Datos personales y bancarios:

Age (numérica): edad de la persona de contacto.

Job (categórica): tipo de trabajo.

Marital (categórica): estado civil.

Education (categórica): educación.

Housing (categórica): tiene prestamo hipotecario.

Loan (categórica): tiene algún préstamo personal.

b) Último contacto de la campaña actual :

Contact (categórica): tipo de comunicacion.

Month (categórica): mes de contacto.

Day (categórica): día de la semana.

Duration (numérica): duracion de la llamada

c) Otros datos de interés:

Campaign (numérica): número de contactos realizados

Pdays (numérica): número de días pasados de la última campaña realizada.

Previous (numérica): número de contactos previo a esta campaña.

Poutcome (categórica): resultado de la campaña de marketing anterior.

d) Contexto económico y social:

emp.var.rate(numérica): tasa de variación de empleo.Indicador cuartílico

cons.price.idx (numérica): índice de precio del consumidor.Indicador mensual

cons.conf.id (numérica): indice mensual de confianza del consumidor

euribor3m (numérica): interés Euribor a 3 meses. Indicador diario en porcentaje

nr.employees (numérica): número de empleados

Variable objetivo (binaria):

y: subscripcion del cliente a plazo fijo: ['yes','no']

2.3. Posibles Valores de las Variables Categóricas

A continuación se lista todos los posibles valores de las variables categóricas para realizar posteriormente un estudio de su tabla de frecuencias y sacar las primeras conclusiones.

In [17]:

```
# Listado de posibles valores en categóricas.
# 1. Se recorre todas las columnas del dataframe
for col in df.columns:
   # 2. Si es categórica se imprime los posibles valores
    if df[col].dtype == 'object':
        # 3. Se obtiene los valores mediante método unique
        print(f'Column {col} values: {df[col].unique()}')
```

```
Column job values: ['housemaid' 'services' 'admin.' 'blue-collar' 'techni
cian' 'retired'
 'management' 'unemployed' 'self-employed' 'unknown' 'entrepreneur'
 'student']
Column marital values: ['married' 'single' 'divorced' 'unknown']
Column education values: ['basic.4y' 'high.school' 'basic.6y' 'basic.9y'
'professional.course'
 'unknown' 'university.degree' 'illiterate']
Column default values: ['no' 'unknown' 'yes']
Column housing values: ['no' 'yes' 'unknown']
Column loan values: ['no' 'yes' 'unknown']
Column contact values: ['telephone' 'cellular']
Column month values: ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar' 'ap
r' 'sep']
Column day_of_week values: ['mon' 'tue' 'wed' 'thu' 'fri']
Column poutcome values: ['nonexistent' 'failure' 'success']
Column y values: ['no' 'yes']
```

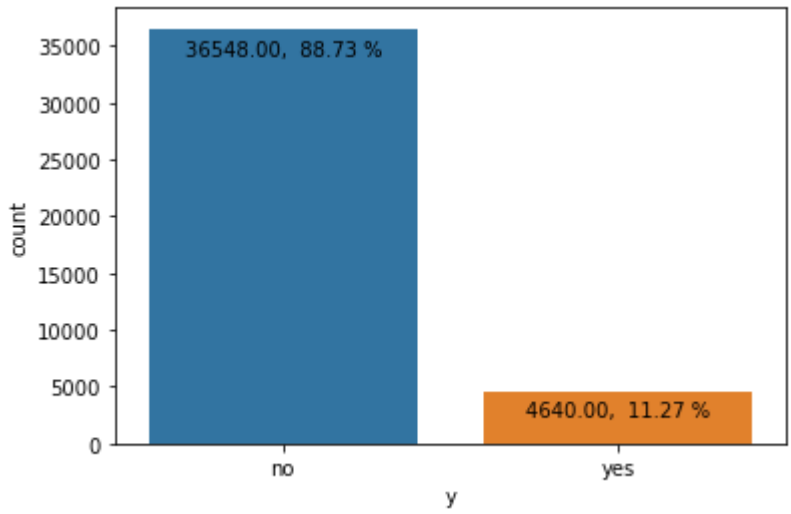
2.4. Balance del dataset

Como parte del estudio se analiza el balance del conjunto de datos que consiste en medir el porcentaje de datos que coresponde a cada caso y saber si el conjunto está balanceado o no. Conjunto de datos balanceados favorecen la predicción frente a conjuntos desbalanceados.

In [18]:

```
# 1. Libreria para importar otras hojas ipynb
#!pip install import_ipynb
import import_ipynb
# 2. Se importa nuestra librería de funciones auxiliares
import functions
# 3. Se importa librería para el cálculo del balanceo de datos
import seaborn as sns
# 4. Se importa para imprimir la gráfica de balanceo.
import matplotlib.pyplot as plt
# 5. Se llama a la hoja de funciones auxiliares
%run functions.ipynb
# 6. Se captura el balance en una variable ax
# mediante countplot de seaborn
ax = sns.countplot(x = df["y"])  #Imbalanced dataset
# 7. Calcula porcentaje de cada valor del histograma
label_values(ax, spacing=-15)
# 8. Se muestra gráfico
plt.show()
```

```
importing Jupyter notebook from functions.ipynb
```



Se observa que hay un 88.73% de clientes que no se subscriben frente a un 11.27% que si. Este reparto hace que el dataset no esté balanceado y será susceptible de aplicar técnicas de balanceo o remuestreo que se implementará más adelante

3. Limpieza y renombrado de variables

Para una mayor claridad se realiza un renombrado de las variables

In [19]:
```
# 1. Se renombran las columnas con rename
df = df.rename(columns={'y': 'deposit','cons.price.idx':'ipc','default':
                        'duration':'call_duration','campaign':'n_contacts
                         'poutcome':'previous_results', 'pdays': 'days_fr
                         'nr.employed':'n_employed','emp.var.rate':'emp_v
# 2. Se visualiza para su comprobación.
df.head(5)
```

Out[19]:

	age	job	marital	education	debts	mortgage	loan	contact	month	day_of_
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	57	services	married	high.school	unknown	no	no	telephone	may	
2	37	services	married	high.school	no	yes	no	telephone	may	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	
4	56	services	married	high.school	no	no	yes	telephone	may	

5 rows × 21 columns

4. EDA

La exploración de datos para su análisis EDA consiste en el estudio de tablas de frecuencia de los diferentes datos así como el estudio de datos ausentes como fuera de rango. Con la finalidad de realizar un análisis concluyente de caracterización de datos (inteligencia de datos) que permita un correcto procesado como paso previo para su modelización y posterior predicción.

4.1. Distribucion de tipos y Missing Values

Selección de variables categóricas y numéricas

En el siguiente bloque se realiza la separación de tipo de variables categóricas y numéricas. Se obtiene un balance de 10 numéricas y 11 categóricas donde se incluye la variable objetivo subscripción del cliente a plazo fijo

In [20]:
```
# Separacion de categóricas y numéricas con select_dtypes :

# 1. Se obtiene el conjunto de numéricas
df_numerical = df.select_dtypes(exclude="object")
# 2. Se obtiene el conjunto de categóricas
df_catgorical = df.select_dtypes(include="object")
# 3. Se imprime por pantalla el listado de numéricas y categóricas
print('Variables numéricas {} '.format(df_numerical.columns))
print('Variables categóricas {} '.format(df_catgorical.columns))
```

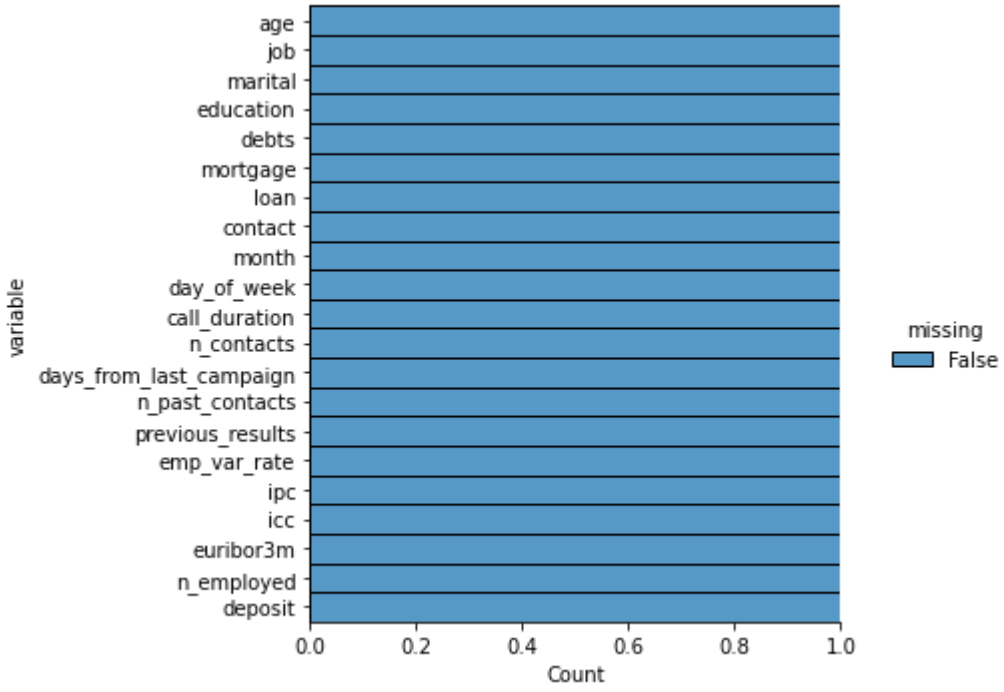
```
Variables numéricas Index(['age', 'call_duration', 'n_contacts', 'days_fr
om_last_campaign',
       'n_past_contacts', 'emp_var_rate', 'ipc', 'icc', 'euribor3m',
       'n_employed'],
      dtype='object')
Variables categóricas Index(['job', 'marital', 'education', 'debts', 'mor
tgage', 'loan', 'contact',
       'month', 'day_of_week', 'previous_results', 'deposit'],
      dtype='object')
```

Detección y tratamiento de valores missing

En esta celda se analiza los valores ausentes para imputar dichos datos o eliminarlos. Se observa por la gráfica obtenida que no hay valores missing y se interpreta que ya se ha imputado en el registro de datos como "unknown" aquellos valores missing o ausentes.

In [22]:
```
# 1. Se dimensiona la figura a plotear
plt.figure(figsize=(10,6))
# 2. Se plotea para cada variable si hay un valor missing
sns.displot(
    data=df.isna().melt(value_name="missing"),
    y="variable",
    hue="missing",
    multiple="fill",
    aspect=1.25)
```

Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x223bce98af0>

<Figure size 720x432 with 0 Axes>
```

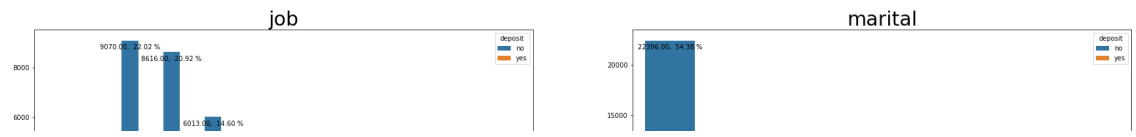


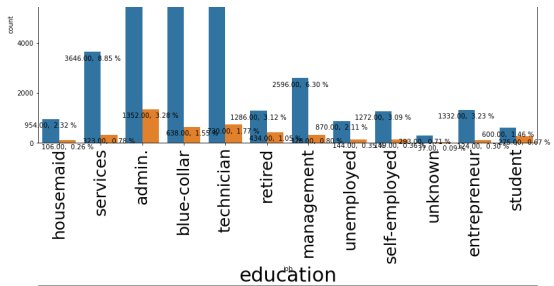
4.2. Tabla de frecuencias para variables categóricas

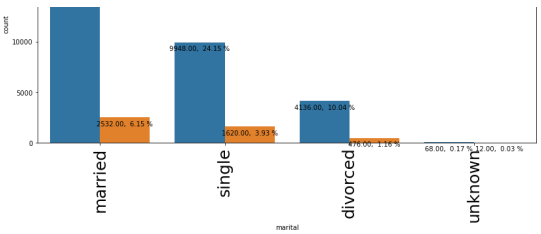
Se recorre cada una de las variables categóricas, para obtener el balance o histograma de cada uno de sus valores en función de la variable objetivo. Además se obtiene el porcentaje del total de cada valor. A continnuación se puede observar las gráficas obenidas. Esto nos permite saber cóo se distribuye los datos y sacar conclusiones inteeresantes para el negocio de la entidad bancaria.

In [23]:

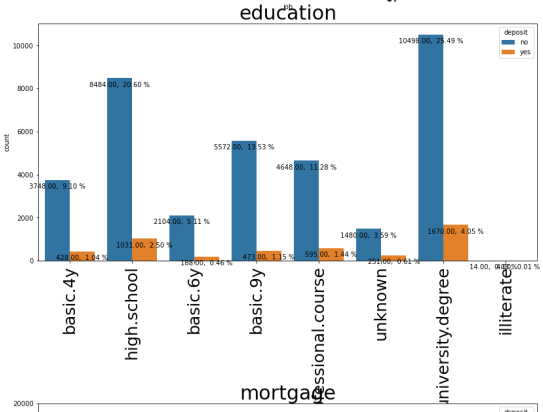
```
#Tabla de frecuencias para la variables categóricas

# 1. Ya se importó la librería para el plot de gráficas
#import matplotlib.pyplot as plt

# 2. Se dimensiona la figura a plotear
fig = plt.figure(figsize = (30, 60))

# 3. Se recorre todas las variables categóricas
for idx, i in enumerate(df_catgorical.columns):
    # 4. Para cada variable categórica, se le asigna un un subplot secue
    fig.add_subplot(6, 2, idx+1)

    # 5. Se contabiliza el valor de las variables en función del target
    ax=sns.countplot(x = df_catgorical.iloc[:, idx],hue=df["deposit"])
    # 6. Se dibuja el porcetaje y valor de contabilizacion de las variab
    label_values(ax, spacing=-15)
    # 7. Se realiza ajustes y etiquetado de cada subplot como título, no
    plt.subplots_adjust(hspace=0.6)
    locs, labels = plt.xticks(fontsize=25)
    plt.setp(labels, rotation=90)
    plt.title(i,fontsize=30)
# 8. Se plotea la gráfica resultante
plt.show()
```



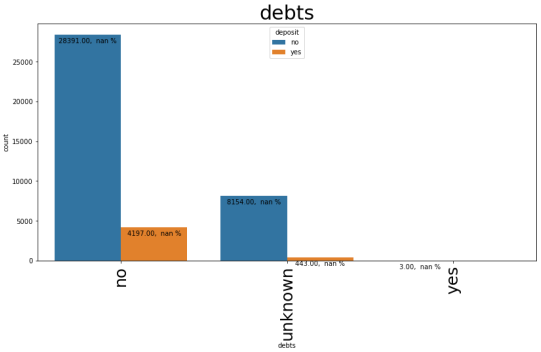




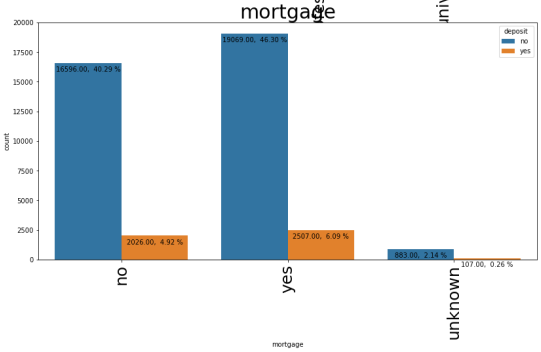
education



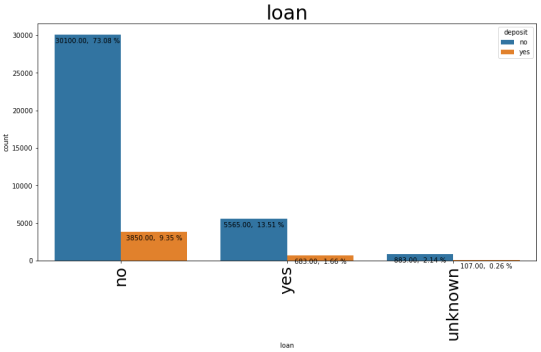
debts



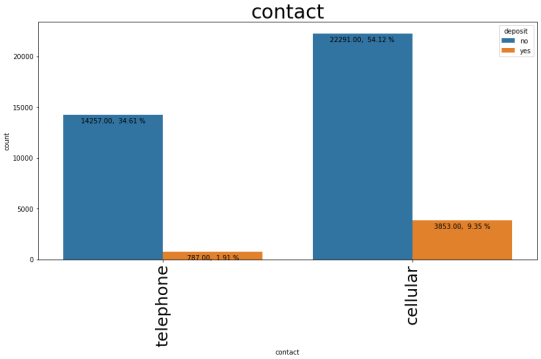
mortgage



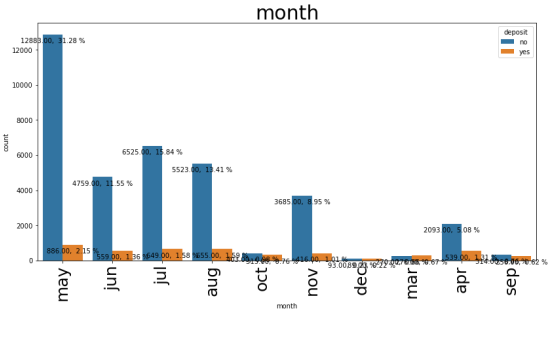
loan



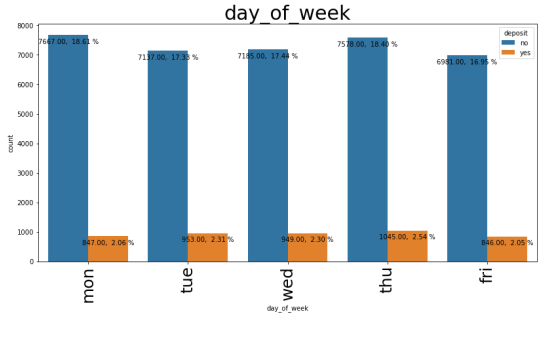
contact



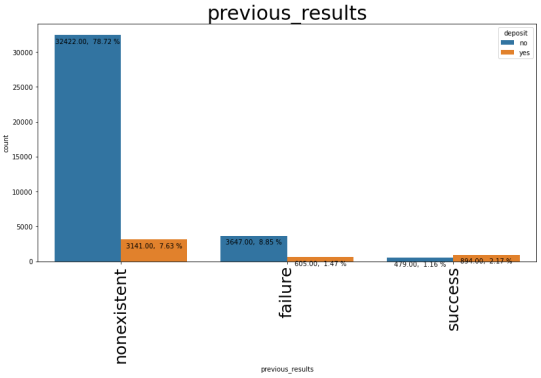
month



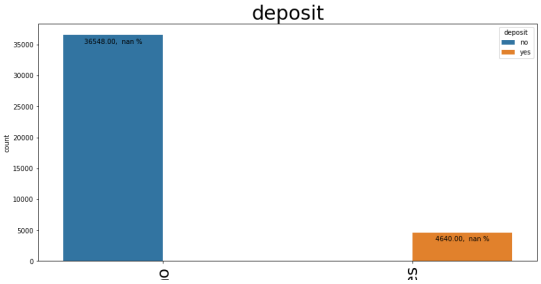
day_of_week



previous_results



deposit



n deposit ye

Pocentaje para cada valor

La gráficas anteriores muestran un porcentaje del total pero para saber dentro de cada uno de los valores de cada variable, qué porcentaje o proporción de subscriptores solicitaron un depósito a plazo fijo, es necesario calcularlo . Para ello se ejecuta el siguiente bloque donde se lista la proporción de subscriptores para cada una de los valores de cada variable. Además se calcula quienes fueron en proporción, los que solicitaron más dicho depósito.

In [24]:

```
# Porcentaje de subscriptores para cada valor de cada de las variables c
# 1. Recorrido del vector de variables categóricas
for i in df_catgorical.columns:
    max=0
    cat=''
    # 2. Recorrido del listado de posibles valores de una variable categ
    for x in df[i].unique():

        # 3. Subconjunto de datos que contienen ese valor
        df_type_cat = df[df[i] == x]
        # 4. Agrupación del subconjunto por el resultado de subscripción
        df_analisys = df_type_cat.groupby('deposit')

        # 5. Cálculo de porcentajes por cada grupo.
        #print(type(df_analisys.size()))
        #print(df_analisys.size())
        sizes=df_analisys.size()
        #print(type(sizes))
        #print(type(sizes.size))

        # 6. Comprueba si el máximo para actualizar el máximo valor de po
        if(sizes.size == 2):
            total = sizes[0] + sizes[1]

            porcentage = (sizes[1]/total)*100
            if(porcentage > max):
                max=porcentage
                cat = x
            # 7. Muestra el porcentaje de subscriptores para cada valor
            print(f'Porcentage {x} values: {porcentage}')

    #8.  Muestra quienes solicitaron más dicho depósito para una variabl
    print(f'Porcentage for {i} max {cat} values: {max}')
```

```
Porcentage housemaid values: 10.0
Porcentage services values: 8.138070042831949
Porcentage admin. values: 12.972558050278257
Porcentage blue-collar values: 6.894315971471795
Porcentage technician values: 10.826041821147857
Porcentage retired values: 25.232558139534884
Porcentage management values: 11.21751025991792
Porcentage unemployed values: 14.201183431952662
Porcentage self-employed values: 10.48557353976073
Porcentage unknown values: 11.212121212121213
Porcentage entrepreneur values: 8.516483516483516
Porcentage student values: 31.428571428571427
Porcentage for job max student values: 31.428571428571427
Porcentage married values: 10.157252888318355
Porcentage single values: 14.004149377593361
```

```
Porcentage divorced values: 10.320901994796184
Porcentage unknown values: 15.0
Porcentage for marital max unknown values: 15.0
Porcentage basic.4y values: 10.24904214559387
Porcentage high.school values: 10.835522858644246
Porcentage basic.6y values: 8.202443280977311
Porcentage basic.9y values: 7.82464846980976
Porcentage professional.course values: 11.348464619492656
Porcentage unknown values: 14.500288850375506
Porcentage university.degree values: 13.724523339907954
Porcentage illiterate values: 22.22222222222222
Porcentage for education max illiterate values: 22.22222222222222
Porcentage no values: 12.878973855406898
Porcentage unknown values: 5.152960335000581
Porcentage for debts max no values: 12.878973855406898
Porcentage no values: 10.879604768553323
Porcentage yes values: 11.619391916944753
Porcentage unknown values: 10.808080808080808
Porcentage for mortgage max yes values: 11.619391916944753
Porcentage no values: 11.34020618556701
Porcentage yes values: 10.931498079385403
Porcentage unknown values: 10.808080808080808
Porcentage for loan max no values: 11.34020618556701
Porcentage telephone values: 5.231321457059293
Porcentage cellular values: 14.737607099143208
Porcentage for contact max cellular values: 14.737607099143208
Porcentage may values: 6.434744716391895
Porcentage jun values: 10.511470477623167
Porcentage jul values: 9.046557011430165
Porcentage aug values: 10.60213661379087
Porcentage oct values: 43.871866295264624
Porcentage nov values: 10.14386734942697
Porcentage dec values: 48.9010989010989
Porcentage mar values: 50.54945054945055
Porcentage apr values: 20.47872340425532
Porcentage sep values: 44.91228070175438
Porcentage for month max mar values: 50.54945054945055
Porcentage mon values: 9.948320413436692
Porcentage tue values: 11.779975278121137
Porcentage wed values: 11.667076469141874
Porcentage thu values: 12.118752174417256
Porcentage fri values: 10.80873898045228
Porcentage for day_of_week max thu values: 12.118752174417256
Porcentage nonexistent values: 8.83221325534966
Porcentage failure values: 14.22859830667921
Porcentage success values: 65.1128914785142
Porcentage for previous_results max success values: 65.1128914785142
Porcentage for deposit max  values: 0
```

Conclusiones:

1. Trabajo: el número más alto (alrededor del 25%) de solicitudes proviene del tipo de trabajo administrador. Sin embargo el estudiante es el que contrata más en proporción dicho servicio y a continuación gente jubilada.
2. Conyugal: Alrededor del 60% de los clientes a los que se contactó estaban casados. Sin embargo los que más contrataron fueron los solteros, posiblemente influenciados porque no tengan préstamo hipotecario.
3. Educación: Se acercó más a los clientes con título universitario y bachillerato en comparación con otros y también tienen una mayor tasa de éxito. (en términos de

número de depósito a plazo). Posiblemente pueda ser buena idea agrupar basic4 y 6 como 'primary school'; y 9 y high school como 'secondary school'

4. Vivienda: El préstamo para vivienda no tiene mucho efecto sobre el número de depósitos a plazo adquiridos.
5. Préstamo: Nos acercamos al 84% de los clientes que no tienen préstamo personal. Aunque parece que no tiene mucho efecto predictivo. Se podría en principio quitar.
6. Contacto: Alrededor del 64% de las llamadas son desde celular y además fueron los que contrataron en mayor medida.
7. Mes: Se sondeó mucho en verano y en Noviembre. Alrededor del 33% se contactó en mayo y en enero y febrero no tenemos datos o no se contactó a nadie. La tasa de éxito fue baja y casi la misma en Junio, Julio, Agosto y Noviembre. Y tasas muy altas en Marzo, Septiembre y Octubre.
8. day_of_week: Tenemos valores recolectados de 5 días. No hay diferencias significativas en el número de clientes abordados y el número de personas suscritas.
9. poutcome: Si un cliente tomó el depósito a plazo la última vez, hay mayores posibilidades de que ese cliente se suscriba nuevamente.

   Resumen: tasa de mayor éxito para cliente solteros, con celular, administradores/retirados/estudiantes que tienen una titulación universitaria, con préstamo hipotecario y sin deudas.

In [ ]:
```
Una forma de visualizar la tabla de totales para una variable en concret
especifica la variable independiente y la variable objetivo. A modo de e
```

In [25]:
```
# Matriz de totales para la variable 'debts'
# 1. Se invoca función crosstab de pandas con variable target y variable
pd.crosstab(df['debts'], df.deposit)
```

Out[25]:

deposit	no	yes
debts		
no	28391	4197
unknown	8154	443
yes	3	0

4.3. Tabla de frecuencias para variables numéricas

A continuación se obtienen para las variables numéricas: los valores estadísticos, la correlación entre ellas y las gráficas con su distribución y balance de las mismas. Este análisis nos ayuda a sacar conclusiones muy valiosas para el negocio en base a esta distribución.

Estadísticas

La tabla de estadísticas refleja cómo se distribuyen los datos del conjunto. En la siguiente tabla se refleja para cada variable: el numero total de datos (41188), la media de cada una de las variables, la desviación estandar que nos da una idea de la dispersión de datos para un porcentaje de datos, el rango de valores entre el mínimo y máximo y por

último los percentiles o cuartiles con la distribución de valores dentro del rango especificado. A continuación se observa el resultado de la tabla de estadísticas de las variables numéricas

In [26]:
```
# Tabla de estadísticas
# 1. Con describe se obtiene la tabla de estadísticas
df_numerical.describe().round().T.style.bar(subset=['mean'], color='#205
                                .background_gradient(subset=['std'], cmap='R
                                .background_gradient(subset=['50%'], cmap='c
```

Out[26]:

	count	mean	std	min	25%
age	41188.000000	40.000000	10.000000	17.000000	32.000000
call_duration	41188.000000	258.000000	259.000000	0.000000	102.000000
n_contacts	41188.000000	3.000000	3.000000	1.000000	1.000000
days_from_last_campaign	41188.000000	962.000000	187.000000	0.000000	999.000000
n_past_contacts	41188.000000	0.000000	0.000000	0.000000	0.000000
emp_var_rate	41188.000000	0.000000	2.000000	-3.000000	-2.000000
ipc	41188.000000	94.000000	1.000000	92.000000	93.000000
icc	41188.000000	-41.000000	5.000000	-51.000000	-43.000000
euribor3m	41188.000000	4.000000	2.000000	1.000000	1.000000
n_employed	41188.000000	5167.000000	72.000000	4964.000000	5099.000000

Observaciones:

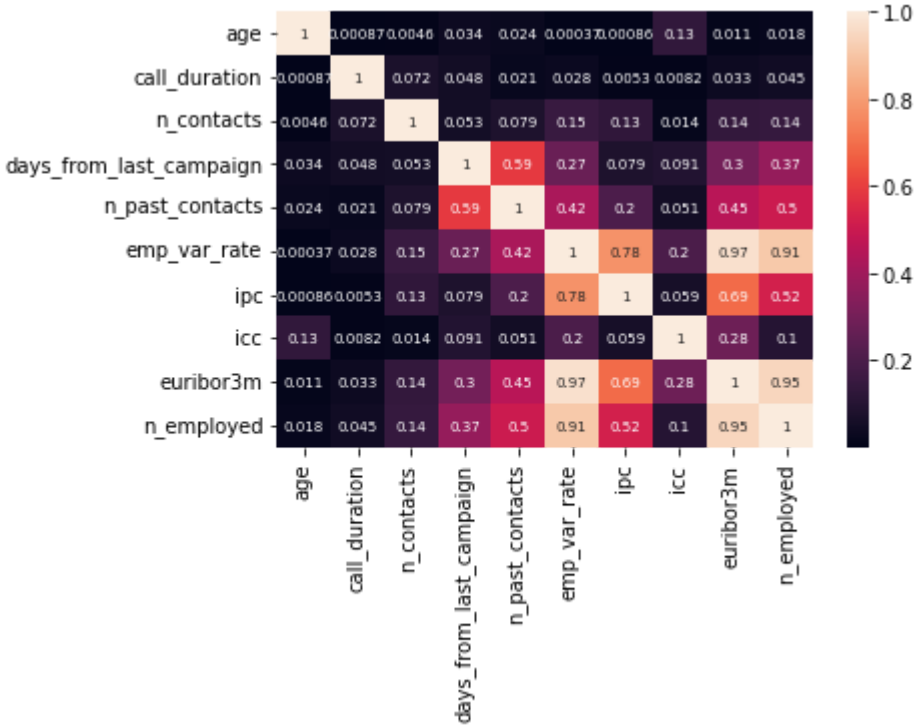
1. Escala de datos diversa: debe normalizarse los datos
2. pdays o days_from_last_campaign debe analizarse en detalle. Parece que 999 necesita ser reemplazado por 0.

Correlacion

La correlación nos da una idea de cómo están relacionadas las variables entre sí. Para ello se constuye dicha matriz y se representa mendiante un heatmap. En la siguiente figura se observa dicho hetamap de la matriz de correlaciones.

In [27]:
```
# Hetamap de la matriz de correlaciones
# 1. Se imorta la librería seaborn para la matriz de correlaciones de  l
import seaborn as sns
# 2. Mediante método corr() se obtiene la matriz de correlaciones entre
cor_matrix = df_numerical.corr().abs()
# 3. Se realiza un heatmap de dicha matriz.
sns.heatmap(cor_matrix , annot= True,annot_kws={"size": 7})
# NOTA: se observa que la tasa de variacion de empleo está relacionada c
```

Out[27]:
```
<AxesSubplot:>
```



Filtrado de algunas variables numéricas:

1. Tasa de variación del empleo (emp_var_rate) y número de empleados (n_employed) se correlacionan positivamente con la tasa de interés Euribor (euribor3m). Por tanto las dos primeras se eliminan porque con el Euribor es más que suficiente como indicador del precio del dinero.

In [28]:
```
# Filtrado de aquellas variables numéricas correladas: emp_var_rate y n_
# 1. Con drop se inidica que variables se quieren eliminar del dataframe
df.drop(['emp_var_rate', 'n_employed'], axis = 1, inplace=True)
#2. Se actualiza el conjunto de variables numéricas
df_numerical = list(set(df_numerical) - set(['emp_var_rate', 'n_employed
```

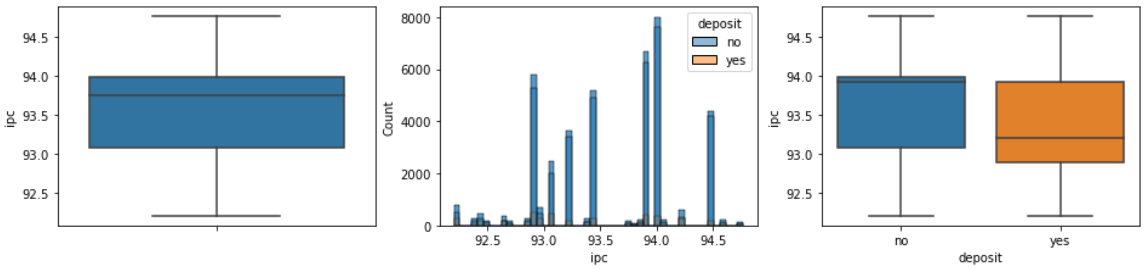
Gráficas de frecuencia

En este presente bloque, se obtiene para cada variable numérica el histograma de valores en función de la variable objetivo y box-plot o diagrama de cajas. Una que se analiza las gráficas resultantes y analizar más en detalle algunas variables, se obtienen las conclusiones que van a continuación.

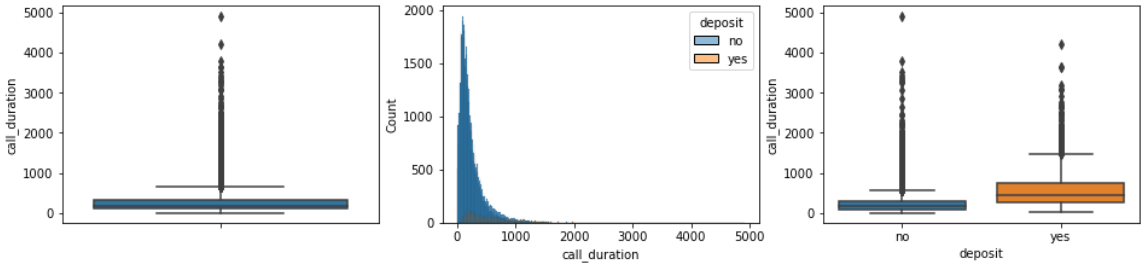
In [29]:
```
# Gráficas de frecuencia
# 1. Indice para la ubiación de los subplot
i = 1
# 2. Recorrido de las vbles numéricas
for column in df_numerical:
    # 3. Título de la vble
    print(column.title())
    # 4. Dimensionado del subplot
    plt.subplots(figsize=(16, 35))
    plt.subplot(len(df_numerical) + 1, 3, i)
    # 5. Box-plot de la vble
    sns.boxplot(y = df[column])
    i += 1
    # 6. Dimensionado e histograma en base a la vble objetivo.
    plt.subplot(len(df_numerical) + 1, 3, i)
```

```
    #sns.distplot(x = df[column])
    sns.histplot(x = df[column])
    #sns.displot(x = column, hue='deposit', data=df)
    sns.histplot(x = df[column], hue='deposit', data=df)
    i += 1
    # 8. Dimensionado y box-plot en base al target
    plt.subplot(len(df_numerical) + 1, 3, i)
    sns.boxplot(x = df["deposit"], y = df[column])
    i += 1
    plt.show()
```

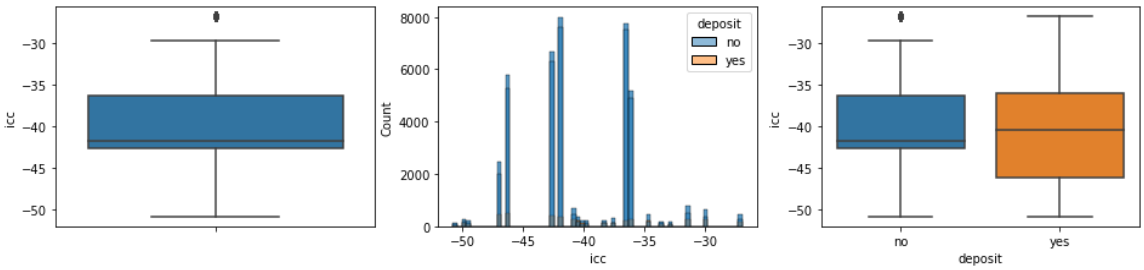
Ipc



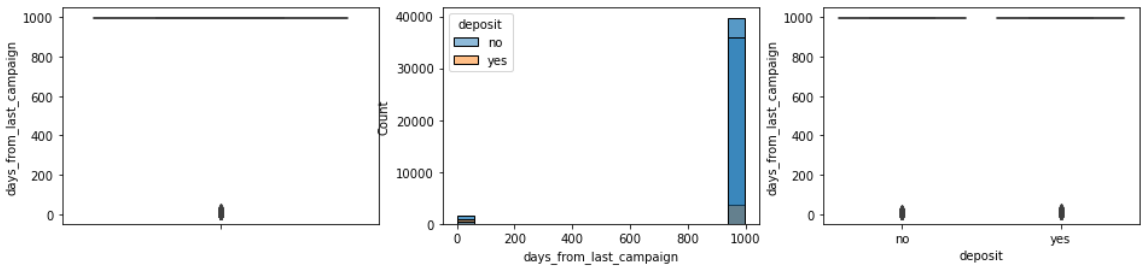
Call_Duration



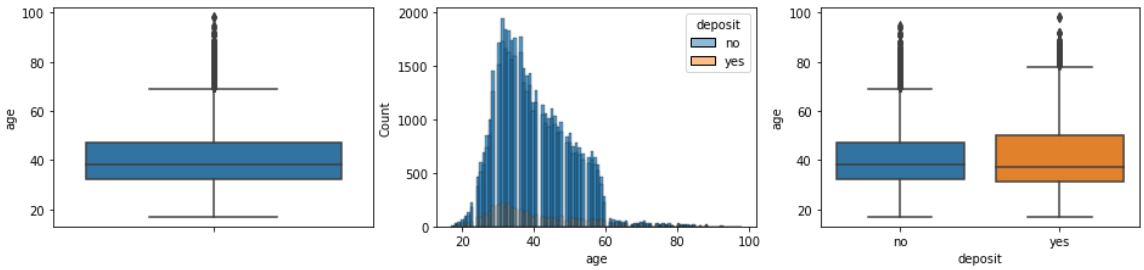
Icc



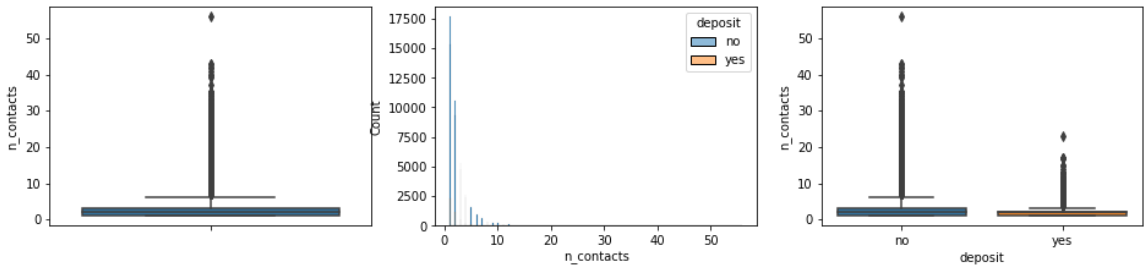
Days_From_Last_Campaign



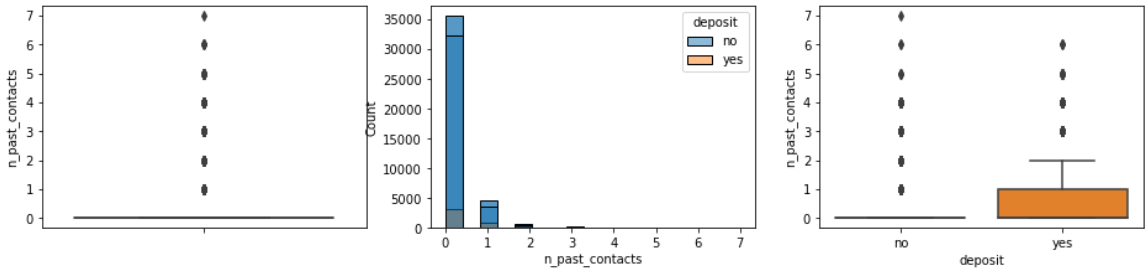
Age



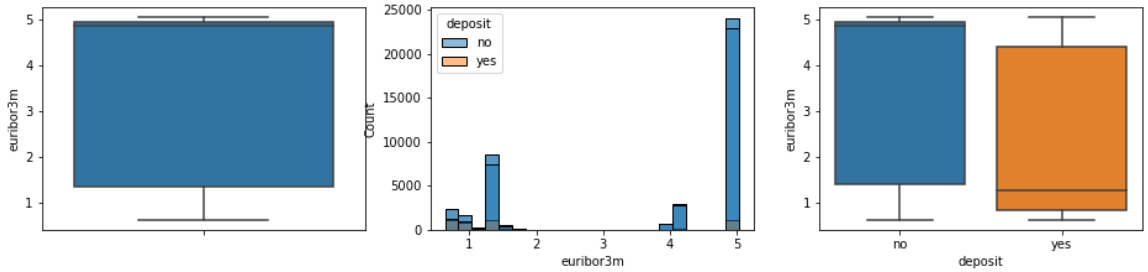
N_Contacts



N_Past_Contacts



Euribor3M



Observación de algunas vbles

Contactos realizados

En el caso de la variable numero de contactos, se observa que cuartil o percentil 95% se corresponde con el umbral 7. Además se observa que por encima de 8 contactos hay una tasa de éxito del 4 % . Por lo tanto, la entidad bancaria o el departamento de Negocios debe saber que a partir de un numero alto de contactos (entorno a 7 u 8), la tasa de éxito de subscripción es muy baja (en torno al 4%).

In [30]:
```
# Cálcuco de precentil 95 y número de casos por debajo de ese percentil
# 1. Se calcula el precentil mediante la función quantile y con len el n
print(df["n_contacts"].quantile(0.95), len(df[df["n_contacts"]<df["n_con
```

```
7.0 38782
```

In [31]:
```
# Tasa de éxito por encima de 8 contactos.
# 1. Cálculo del número de casos que se subscribieron por encima de 8 co
df1 = df[(df["n_contacts"] > 8) & (df["deposit"] == 'yes')]
# 2. Cáculo del porcentaje de éxito por encima de ese umbral
print(len(df1[['n_contacts', 'deposit']]), len(df[df["n_contacts"]>8]))
#df["campaign"] = df["campaign"].apply(lambda x: 8 if x > 8 else x) ???
```

```
56 1377
```

Contactos de otras campañas

A continuación, se obtiene un listado con la tasa en porcentaje de subscriptores para el total de pasados contactos. Se deduce que cuantos más contactos más probabilidad de

subscripción existe.

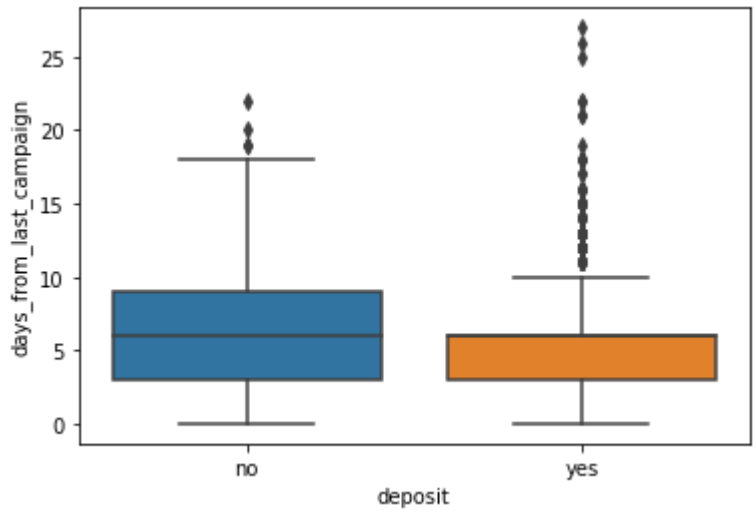
In [32]:
```
# Listado de tasa de éxito de subscriptores en función del número de con
#for i in range(1, max(df["n_past_contacts"])+1):

#1 . Recorrido del número de pasados contactos
total = df['n_past_contacts'].max()+1
for i in range(1, total):
    print(i, end = " ")
    # 2. Balance de subscriptores basado en número de contactos pasados.
    df1 = df[(df["n_past_contacts"] >= i) & (df["deposit"] == 'yes')]
    print(len(df1[['n_past_contacts', 'deposit']]), len(df[df["n_past_co
```

```
1 1499 5625
2 532 1064
3 182 310
4 54 94
5 16 24
6 3 6
7 0 1
```

Días pasados con respecto a la última campaña.

Por las gráficas anteriores de esta variable, se observa que el valor mayoritario es 999 y que se interpreta que se corresponden con aquellos clientes nuevos donde no ha existido campañas anteriores. Si se eliminan dichos casos y se obtiene el diagrma box-plot, se puede observar l adistribución de valores en base al target.

In [33]:
```
# Look into the pdays values which are less than 999
df1 = df[~(df["days_from_last_campaign"] >= 999)]
sns.boxplot(x = df1["deposit"], y = df1['days_from_last_campaign'])
plt.show()
```



Conclusiones:

1. Indice mensual de confianza (icc): a mayor índice de confianza hay mayor probabilidad de éxito para subscribir dicho depósito.
2. Edad (Age): El perfil de contacto se corresponde con mediana edad (38 años) , siendo el 50 % con edades comprendidas entre 32-47. Y la distribución general muestra edades comprendidas entre 25-50.

3. Indice de Precios al Consumidor (ipc) : si este valor es alto, la probabilidad de que el cliente no se suscriba es ligeramente mayor.
4. Numero de contactos o campaña: la mayor parte de los clientes no se ha contactado anteriormente encontrándose en un rango entre 0-10.El 95 % está por debajo de los 8 contactos. Por encima de los 8 contactos se comprueba que hay tasa de éxito mucho menor, entorno al 4% de éxito.
5. Call_duration tiene un valor alto predictivo. Mayor tasa de exito cuanto más dure la llamada.Sin embargo se debe eliminar para obtener un modelo realista predictivo ya que no se sabe hasta que no se produce tal llamada. Además si la duracion es 0 el resultado es no
6. Tasa de Euribor a 3 meses (Euribor3M): si la tasa es alta hay más posibilidades de que los clientes no se subscriban al depósito a plazo.
7. Días desde la última campaña: en un gran número de clientes con valor 999, lo que todo parece indicar que se corresponden con los que no participaron en campañas anteriores. Si se filtra dicho valor se observa en el gráfico anterior, que existen más posibilidades de éxito cuantos menos días pasen. Candidata a eliminarla o al menos imputar el valor 999 a 0.
8. Contactos pasados (n_past_contacts): la mayor parte de los clientes no participaron en otras camapañas, pero aquellos que sí hay una alta probabilidad de que se subscriban.

NOTA: en base al análisis realizado, se procede a eliminar la variable call_duration y para la variable 'days_from_last_campaign'se imputa aquellos valores 999 por 0.

In [35]:
```
# Eliminación de call_duration
# 1. Se procede a eliminar call_duration mediante drop
df.drop(['call_duration'], axis = 1, inplace=True)
# 2. Se actualiza las variables numéricas
#df_numerical = list(set(df_numerical) - set(['call_duration']))
df_numerical = list(set(df_numerical) - set(['call_duration']))
```

In [36]:
```
# Se imputa aquellos valores 999 de days_from_last_campaign a cero
# 1. Mediante replace se sustituye el valor 999 por 0 para la vble 'days_
df['days_from_last_campaign'] = df['days_from_last_campaign'].replace(99
```

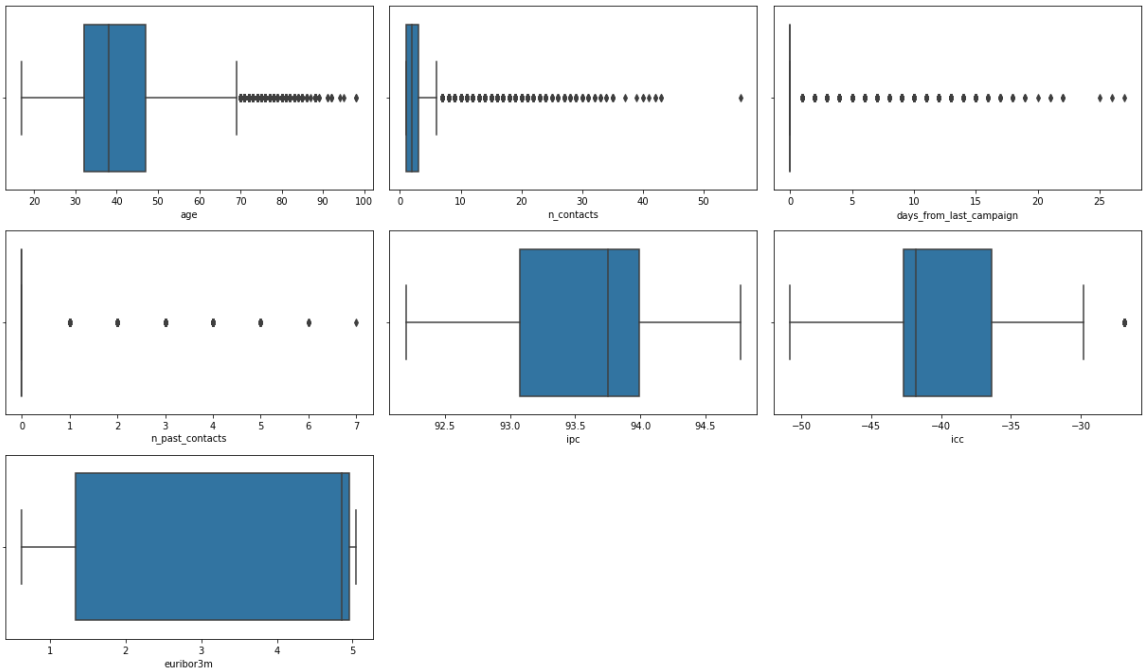
NOTA: Hay una operativa de profiling para EDA que se puede utilizar, dando un balance detallado además de la relación entre sus variables. Se invoca mediente el comando "import pandas_profiling as pp"

4.4. Outliers

Son aquellos valores que están muy por encima de la media. Tres veces la desviacion con respecto a la media tanto hacia arriba como hacia abajo. A continuación se calcula y se muestra visualmente para cada variable numérica.

In [37]:
```
# Outliers de las vbles numéricas
c = 1
# 1. Se filtran el conjunto de vbles numéricas
var_num = df.select_dtypes(include=['float64', 'int64'])
# 2. Se calcula las filas a plotear para tres columnas
```

```
nrows = round(len(var_num.columns) / 3) + 1
# 3. Se dimensiona el ploteado
fig = plt.figure(figsize=(17,10))
# 4. Se recorre las vbles numéricas
for i in var_num.columns:
    # 5. Se dimensiona y se ejcuta el box-plot
    plt.subplot(nrows, 3, c)
    sns.boxplot(x=i, hue='deposit', data=df)
    c += 1
#6. Ploteado final con ajuste automático.
plt.tight_layout()
plt.show()
```



Del listado de variables numéricas se observa que hay outliers por encima en: 'age', 'contacts', 'n_past_contacts' , 'icc' y 'days_from_last_contact_campaign'. En el caso de 'age' se muestra más abajo qué índices corresponden con outliers (por encima de los 70 años), pero como son edades posibles, no se hace nada. En 'n_contacts' se quita los que están por encima de 50. Se puede luego imputar como la media. En 'days_from_last_campaign' se interpreta que no hay outliers porque el valor 0 representa a aquellos clientes que no corresponden a otras campañas y es practicamente la mayoría. En 'n_past_contacts' se interpreta que no hay outliers porque el valor 0 que es la mayoría representa a los que no han estado en otras campañas. En 'icc' hay 447 ouliers por encima de -27. Se pueden eliminar o imputar por la media. De momento lo dejamos estar.

In [38]:

```
# Outliers en vble 'age'
# 1. Se importa nuestra librería de funciones auxiliares
import import_ipynb
import functions
%run functions.ipynb
# 2. Cálculo de los índices que son outliers mediante funcion propia aux
# Ademas se calcula la media y sigma
wrong_age = outliers_indices('age')
out = set(wrong_age)

print(f'Outliers age is {wrong_age} , lenght: {len(out)}')
# NOTA: A modo informativo se muestra porque son edades reales y posible
```

```
Feature age , media: 40.02406040594348, sigma: 10.421249980934235
Outliers age is Int64Index([27757, 27780, 27800, 27802, 27805, 27808, 278
10, 27811, 27812,
            27813,
            ...
            40965, 40966, 40969, 40982, 40983, 40986, 40996, 41004, 4118
3,
            41187],
           dtype='int64', length=369) , lenght: 369
```

In [40]:

```
# Outliers en 'n_contacts'
# 1. Cálculo de los índices que son outliers mediante funcion propia aux
# Ademas se calcula la media y sigma
wrong_contacts = outliers_indices('n_contacts')
out = set(wrong_contacts)
print(f'Outliers contacts is {wrong_contacts} , lenght: {len(out)}')

# Outliers en 'icc'
# 1. Cálculo de los índices que son outliers mediante funcion propia aux
# Ademas se calcula la media y sigma
wrong_icc = outliers_indices('icc')
out = set(wrong_icc)
print(f'Outliers icc is {wrong_icc} , lenght: {len(out)}')
```

```
Feature n_contacts , media: 2.567592502670681, sigma: 2.7700135429021127
Outliers contacts is Int64Index([ 2189,  2234,  2554,  2589,  2590,  261
3,  2631,  2660,  2735,
             2866,
            ...
            35441, 35525, 35587, 35755, 36884, 37587, 37876, 37893, 4006
3,
            40529],
           dtype='int64', length=869) , lenght: 869
Feature icc , media: -40.50260027191399, sigma: 4.628197856174375
Outliers icc is Int64Index([], dtype='int64') , lenght: 0
```

Otros datos de interés en Outliers

En base a las gráficas anteriores, se puede calcular porcentajes de interés de clientes que están por encima de un cierto umbral fuera de rango. A modo de ejemplo se analizan los siguientes datos obtenidos: . En 'n_past_contacts' por encima de 6, hay un porcentaje muy bajo (0.0024 %) . En 'n_contacts' hay una discontinuidad grande por encima de 45. De hecho se observa que por encima de 50 sólo hay un cliente. . En 'icc' se podría establecer el umbral en -27 según la gráfica obtenida. Los cálculos posteriores reflejan que sólo 447 están en esa situación.

In [41]:

```
# Filtrado de 'n_past_contacts' por encima del umbral  6
len (df[df['n_past_contacts'] > 6] ) / len(df) * 100
```

Out[41]:

```
0.0024278916189181313
```

In [42]:

```
# Filtrado de 'n_contacts' por encima de 50 y visualización del filtrado
df[df['n_contacts'] > 50]
```

Out[42]:

	age	job	marital	education	debts	mortgage	loan	contact	month
4107	32	admin.	married	university.degree	unknown	unknown	unknown	telephone	may

In [43]:
```
# Outliers 'icc' por encima de -27
# 1. Cálculo de índices de 'icc' por encima del umbral -27
idx_to_drop = df[df['icc'] > -27].index
# 2. Longitud del filtrado realizado.
len(idx_to_drop)
```

Out[43]:
```
447
```

In [44]:
```
# 3. Visualización del conjunto de datos cuyo 'icc' está por encima de -
df[df['icc'] > -27]
```

Out[44]:

	age	job	marital	education	debts	mortgage	loan	contact	montl
38154	50	management	married	university.degree	no	yes	no	cellular	oc
38155	37	admin.	single	university.degree	no	yes	no	cellular	oc
38156	59	technician	single	basic.6y	no	no	no	cellular	oc
38157	31	admin.	married	university.degree	no	yes	no	cellular	oc
38158	35	admin.	married	high.school	no	yes	no	cellular	oc
...	...	...	...	...	...	...	...	...	..
38596	69	retired	married	basic.4y	no	yes	yes	cellular	oc
38597	18	student	single	basic.6y	no	no	yes	cellular	oc
38598	59	retired	divorced	basic.4y	no	yes	no	telephone	oc
38599	37	admin.	single	university.degree	no	no	no	cellular	oc
38600	78	retired	divorced	basic.6y	no	no	no	telephone	oc

447 rows × 18 columns

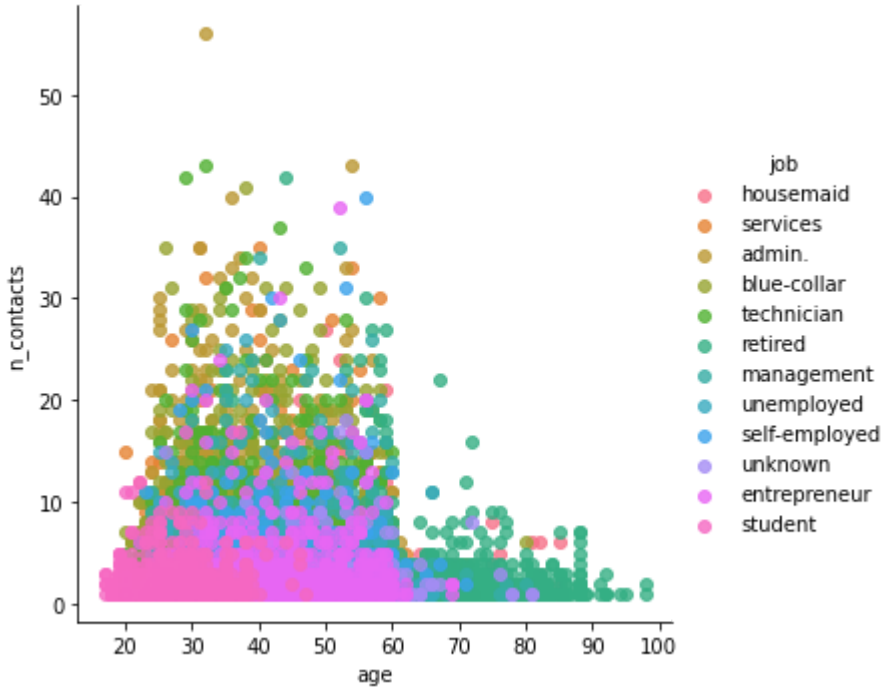
4.5. Otros datos de Interés

Relacion entre variables de entrada

Aunque para el problema de clasificación binaria que nos enfrentamos de contratación de depóstito sería sufieciente. Se podría obtener un análisis todavía más detallado, con la combiancón de varias variables predictoras, pudiéndo extraer más datos en profuncidad. A modo de ejemplo, mediante lmplot se puede extrer la relación entre los contactos del cliente y su edad, así como el tipo de trabajo desempeñado. Dichas conclusiones del estudio, se detallan más abajo.

In [45]:
```
# Gráfica de relacion n_contactos con edad y tipo de trabajo

# 1. Mediante lmplot se plotea las variables referencias
sns.lmplot( x="age", y="n_contacts", data=df, fit_reg=False, hue='job',
```

Out[45]:
```
<seaborn.axisgrid.FacetGrid at 0x223bb1a91f0>
```



Se observa por ejemplo que los estudiantes están comprendidos entre 20-40 años y han recibido menos contactos. Sin embargo los técnicos tiene una franja de edad mayor y han recibido más contactos. Este tipo de gráficos nos da información muy valiosa de nuestra poblacion.

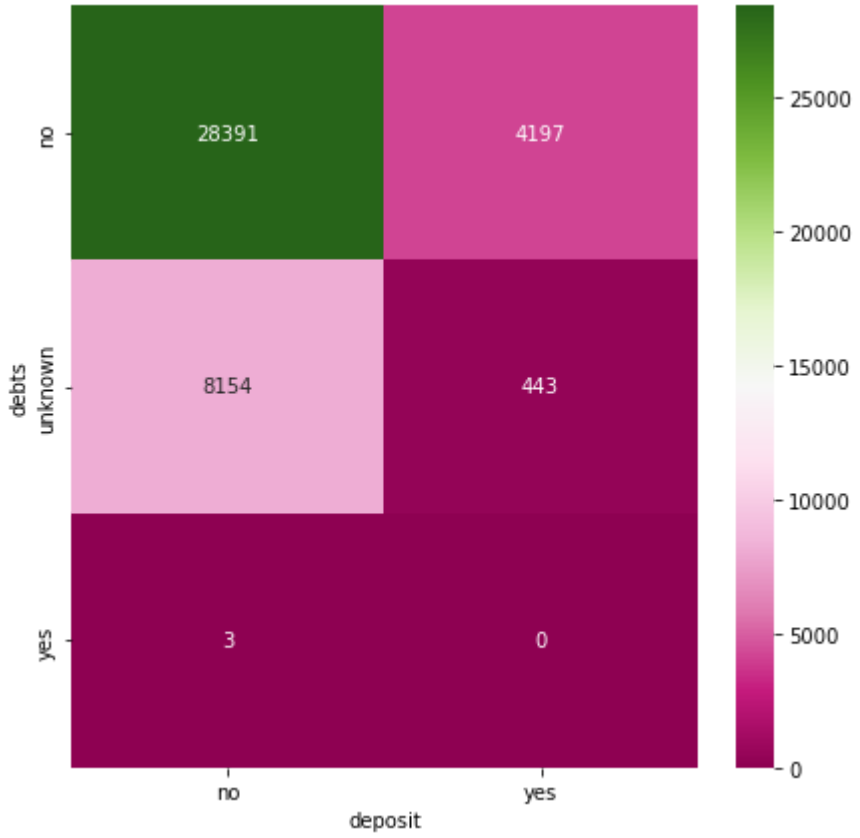
Importancia de variables

Se puede ver la relación de una variable predictora con la target a través del valor p-value que ofrece el método chi_cuadrado y mediante el método crosstab de pandas se puede observar la matriz de resultados. A modo de ejemplo se realiza dicho estudio para la variable debts o default originalmente. En dicho cálculo se puede observar que el p-value está muy por debajo de 0.05, por tanto la relación (correlación) entre la disponibilidad de crédito del cliente y el resultado es estadísticamente significativa.

In [46]:
```
# Obtención del chi-cuadrado de 'debts'

# 1. Se importa paquete stats de la libería scipy
from scipy.stats import chi2_contingency, fisher_exact
# 2. Cáculo de p-value mediante chi2_contingency de su matriz de resulta
p = chi2_contingency(pd.crosstab(df['debts'], df['deposit']))
# 3. Se imprime p-value
print("p-value = " + str(p[1]))

# NOTA: Dado que los datos son categóricos, se usa la prueba de chi-cuad
```

```
p-value = 5.1619579513916376e-89
```

In [47]:
```
# Matriz de resultados de la vble 'debts'
# 1. Dimensionado de la figura a plotear
plt.figure(figsize=(7, 7))
# 2. Heatmap de su matriz de resultados
sns.heatmap(pd.crosstab(df['debts'], df['deposit']), fmt="d", cmap="PiYG
```



5. Preprocesado de Datos

A continuación se procesan los datos a través de la función tranform de la siguiente manera:

a. Se binariza la variable target

b. Se codifican las variables debts, mortgage y loan mediante el método LabelEncoder mediante la numeración {'unknown': 0, 'yes': 1, 'no': 2}

c. Se codifican los meses del 1 al 12 secuencialmente

d. Se codifica las variables contact y previous_results según la siguiente codificación contact --> {'unknown': 0, 'cellular': 1, 'telephone': 2} previous_results --> {'nonexistent':0, 'failure':1, 'success':2}

e. Se codifica el resto de variables como one hot encoding : job, marital, education y day_of_week.

f. Se estandariza todas las variables numéricas con StandardScaler

Dicha función está implementada en nuestra librería de funciones auxiliares propias. Como resultado, se puede observar el dataframe procesado resultante junto con el nombre de sus columnas.

In [48]:

```
# Procesado de datos del dataframe original

# 1. Se importa nuestra librería de funciones auxiliares
import import_ipynb
import functions
```

```
%run functions.ipynb

# 2. Se carga en memoria el dataframe original.
import pandas as pd
url='bank-additional-full.csv'
df_ref = pd.read_csv(url,sep=";")

# 3. Se realiza la transformacion de variables con renombrado de variabl
rename=True
df2 =transform(df_ref,rename)
```

In [49]:

```
# 4. Se visualiza para su comprobación.
df2.head()
```

Out[49]:

	age	n_contacts	days_from_last_campaign	n_past_contacts	ipc	icc	eu
0	1.533034	-0.565922	0.195414	-0.349494	0.722722	0.886447	
1	1.628993	-0.565922	0.195414	-0.349494	0.722722	0.886447	
2	-0.290186	-0.565922	0.195414	-0.349494	0.722722	0.886447	
3	-0.002309	-0.565922	0.195414	-0.349494	0.722722	0.886447	
4	1.533034	-0.565922	0.195414	-0.349494	0.722722	0.886447	

5 rows × 39 columns

In [50]:

```
# 5. Columnas definitivas del dataframe resultante.
df2.columns, len(df2.columns)
```

Out[50]:

```
(Index(['age', 'n_contacts', 'days_from_last_campaign', 'n_past_contact
s',
        'ipc', 'icc', 'euribor3m', 'deposit_n', 'debts_n', 'mortage_n',
        'loan_n', 'month_n', 'contact_n', 'previous_results_n',
        'job_blue-collar', 'job_entrepreneur', 'job_housemaid',
        'job_management', 'job_retired', 'job_self-employed', 'job_servic
es',
        'job_student', 'job_technician', 'job_unemployed', 'job_unknown',
        'marital_married', 'marital_single', 'marital_unknown',
        'education_basic.6y', 'education_basic.9y', 'education_high.schoo
l',
        'education_illiterate', 'education_professional.course',
        'education_university.degree', 'education_unknown', 'day_of_week_
mon',
        'day_of_week_thu', 'day_of_week_tue', 'day_of_week_wed'],
       dtype='object'),
 39)
```

Preparacion de datos en train-test

Por último para la realización del modelado, se dividen los datos en datos de entrenamiento (train) y test. Al no especificar explícitamente el porcentaje de reparto, por defecto es un 75 % de datos utilizados para el entrenamiento y un 25 % para el test. En concreto 30891 para entrenamiento y 10297 para test.

In [51]:

```
# Split train test

# 1. Se importa el paquete train_test_split de sklearn.model_selection
from sklearn.model_selection import train_test_split
```

```
# 2. Separación de los datos de entrada y salida
y = df2['deposit_n']
x = df2.drop('deposit_n', axis = 1)
# 3. Split de los datos en train y test
x_train,x_test,y_train,y_test = train_test_split(x,y, random_state=42)
```

In [52]:

```
#x_train.shape --> 30891
#x_test.shape # --> 10297
x_test

41188
```

Out[52]:

```
41188
```

6. Modelado

Una vez obtenidos los datos procesados de entrenamiento y test, se analizará los modelos más representativos con sus tasas de acierto o precisión y se explicará en detalle los resultados obtenidos así como las característias más importantes de cada uno de ellos.

6.1. Regresion logística

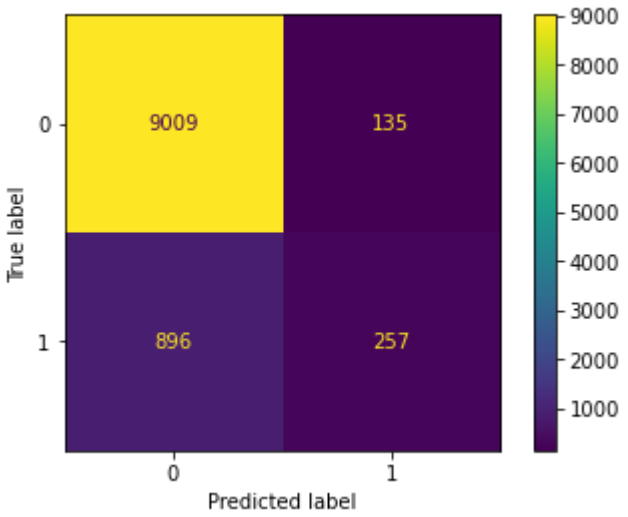
La regresión logística es un modelo matemático en el cual se le aplica el operador logístico (logit) de la probabilidad de que un evento ocurra y está basado en modelos estadísticos lineales fácilmente interpretables. La variable objetivo se predice basándose en una ponderación o coeficiente de las variables independientes o predictoras. Para su implementación, se invoca el modelo 'LogisticRegression' del paquete linear_model de sklearn. Para su validación, se compila o entrena utilizando la validación cruzada repetida con 10 iteraciones, donde se obtiene un 'accuracy' medio de 0.900521 con desviación en sus repreticiones de 0.004396 Sobre el modelo obtenido, se calculan las predicciones de los datos de test, con el objetivo de extraer el informe de clasificación (tasa de acierto y sensibilidad) y la matriz de confusión (casos acertados y falsos positivos/negativos). Finalmente se obtiene la curva ROC. En el siguiente bloque se observa dicho modelo lineal con sus gráficas y análisis posterior de las mismas.

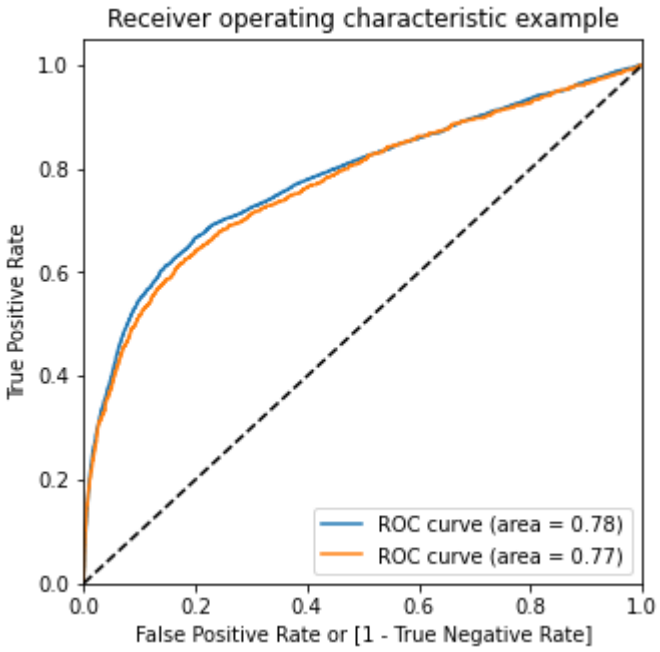
In [53]:

```
# Implementación Modelo logístico
# 1. Se importa nuestra librería de funciones auxiliares
import import_ipynb
%run functions.ipynb
# 2. Se importa librerías necesarias para modelo de regresión logística
# y validación cruzada repetida
from sklearn.linear_model import LogisticRegression
from sklearn import model_selection

# 3. Se importa el resto de librerías: matriz de confusión, curva ROC, i
from sklearn.metrics import plot_confusion_matrix,classification_report,
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
```

```
import matplotlib.pyplot as plt
#from sklearn.metrics import precision_score

# 4. Se invoca al objeto de modelo logístico 'LogisticRegression'
logreg = LogisticRegression(solver='liblinear', random_state = 100, max_
name='Logistic Regression'

# 5. Se compila modelo con validación cruzada repetida con 10 splits y a
kfold = model_selection.KFold(n_splits=10, shuffle = True, random_state=
cv_results = model_selection.cross_val_score(logreg, x_train, y_train, cv
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)
logreg.fit(x_train, y_train)

# 6. Se extrae: informe de clasificación y matriz de consusión.
report(logreg)

# 7. Se extrae la predición del modelo
y_train_pred_lr, y_train_pred_lr_prob, y_test_pred_lr,y_test_pred_lr_prob

# 8. Se dibuja la curva ROC en base a la predicción obtenida
draw_roc(y_train, y_train_pred_lr_prob, y_test, y_test_pred_lr_prob)

# 9. Se imprime el 'acuracy' de entrenamiento y test mediante accuracy_s
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_lr))
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_lr))
```

```
Logistic Regression: 0.900521 (0.004396)
              precision    recall  f1-score   support

           0       0.99      0.91      0.95      9905
           1       0.22      0.66      0.33       392

    accuracy                           0.90     10297
   macro avg       0.60      0.78      0.64     10297
weighted avg       0.96      0.90      0.92     10297
```






```
Accuracy train:  0.900780162506879
Accuracy test:  0.8998737496358162
```

Análisis de Métricas

En el informe de clasificación se observa que el conjunto de test o validación se compone de 9905 casos pertenecientes al no y 392 al sí, por tanto está desbalanceado. Y se observa la precisión con la que se acertó:

1. Precission: 99% de tasa de acierto para la no contratacion frente a un 22% de contratacion de deposito.
2. Recall o ratio verdaderos positivos o sensibilidad: 91 % para el no y 66% para el si.

Además se obtiene la matriz de consfusión con el detalle de los casos postitivos/negativos acertados y los falsos positivos/negativos.

Y se obtiene también la curva ROC con diferentes valores de sensibilidad-especificidad para diferentes puntos de corte, donde se establece una AUC de 0.78, por tanto se consiera un modelo de buena calidad.

Se observa una tasa de acierto en test muy parecida a la de entrenamiento en torno a 0.9

Interpretacion Statsmodel en Regresion Logística

La siguiente tabla resumen es un resumen descriptivo sobre los resultados de la regresión. A través del paquete statsmodel, se invoca a la función logit para la contrucción del modelo de regresión logística. Del cual se obtiene la tabla resumen mediante el método summary. En ella se observa los siguientes parámetros de interés: a. Coeficientes: son los coeficientes de las variables independientes de la ecuación lineal. Representa lo que cambia la salida en un incremento de una unidad de la variable predictora. Si la edad de una persona es 1 unidad más, tendrá una probabilidad de 0.076 unidad menos de subscribirse al depósito. b. Pseudo R-squ: representa la bondad de ajuste del modelo de regresión. Está muy por debajo del 1. c. std err refleja el nivel de precisión de los coeficientes d. p-value: valores por debajo de 0,05 son estadísticamente significativos. e. Intervalo de confianza: representa el rango en el que es probable que

caigan nuestros coeficientes (con una probabilidad del 95 %) f. z: relación entre la estimación del coeficiente y su error estándar. Equivalente al t en regresión lineal.

In [54]:

```
# Tabla resumen de regresión logísticas
# 1. Se importa el paquete api de statsmodel
import statsmodels.api as sm

# 2. Construccion del modelo y compilación del mismo
log_reg = sm.Logit(y_train, x_train).fit()

#3. Impresión de la tabla resumen
print(log_reg.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.282952
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:              deposit_n   No. Observations:
30891
Model:                          Logit   Df Residuals:
30853
Method:                           MLE   Df Model:
37
Date:                Thu, 27 Jan 2022   Pseudo R-squ.:
0.1973
Time:                        07:36:11   Log-Likelihood:                -8
740.7
converged:                       True   LL-Null:                       -1
0889.
Covariance Type:            nonrobust   LLR p-value:
0.000
==============================================================================
                                 coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
age                             -0.0076      0.025     -0.300      0.764
-0.057       0.042
n_contacts                      -0.1648      0.031     -5.356      0.000
-0.225      -0.104
days_from_last_campaign         -0.4221      0.024    -17.477      0.000
-0.469      -0.375
n_past_contacts                 -0.0946      0.028     -3.385      0.001
-0.149      -0.040
ipc                              0.4597      0.026     17.697      0.000
0.409       0.511
icc                              0.3349      0.019     17.244      0.000
0.297       0.373
euribor3m                       -0.9505      0.028    -34.446      0.000
-1.005      -0.896
debts_n                         -0.3746      0.066     -5.675      0.000
-0.504      -0.245
mortage_n                       -0.0343      0.020     -1.691      0.091
-0.074       0.005
loan_n                          -0.0099      0.028     -0.355      0.723
-0.065       0.045
month_n                         -0.0487      0.009     -5.235      0.000
-0.067      -0.030
contact_n                       -1.0888      0.051    -21.508      0.000
```

```
-1.188      -0.990
previous_results_n              -0.3525      0.084     -4.206      0.000
-0.517      -0.188
job_blue-collar                 -0.3752      0.075     -4.978      0.000
-0.523      -0.227
job_entrepreneur                -0.2242      0.125     -1.795      0.073
-0.469       0.021
job_housemaid                   -0.4418      0.151     -2.927      0.003
-0.738      -0.146
job_management                  -0.1691      0.086     -1.960      0.050
-0.338   -4.37e-05
job_retired                      0.0819      0.107      0.764      0.445
-0.128       0.292
job_self-employed               -0.2203      0.117     -1.883      0.060
-0.450       0.009
job_services                    -0.2457      0.084     -2.929      0.003
-0.410      -0.081
job_student                      0.1547      0.114      1.352      0.176
-0.070       0.379
job_technician                  -0.0963      0.069     -1.394      0.163
-0.232       0.039
job_unemployed                  -0.0420      0.127     -0.331      0.740
-0.290       0.206
job_unknown                     -0.3877      0.250     -1.548      0.122
-0.878       0.103
marital_married                 -0.1176      0.063     -1.875      0.061
-0.241       0.005
marital_single                  -0.0766      0.072     -1.066      0.286
-0.218       0.064
marital_unknown                  0.0384      0.453      0.085      0.932
-0.849       0.925
education_basic.6y              -0.1359      0.114     -1.191      0.233
-0.359       0.088
education_basic.9y              -0.2770      0.088     -3.163      0.002
-0.449      -0.105
education_high.school           -0.3087      0.082     -3.756      0.000
-0.470      -0.148
education_illiterate             0.6883      0.763      0.902      0.367
-0.806       2.183
education_professional.course   -0.2546      0.094     -2.703      0.007
-0.439      -0.070
education_university.degree     -0.1655      0.081     -2.049      0.040
-0.324      -0.007
education_unknown               -0.1976      0.116     -1.703      0.089
-0.425       0.030
day_of_week_mon                 -0.2359      0.064     -3.678      0.000
-0.362      -0.110
day_of_week_thu                  0.0295      0.061      0.482      0.630
-0.091       0.150
day_of_week_tue                  0.0059      0.063      0.093      0.926
-0.118       0.129
day_of_week_wed                  0.0461      0.063      0.727      0.467
-0.078       0.170
==============================================================================
========================
```

In [ ]:

```
A continuación se guarda el modelo para luego descargarlo para comprobac
es similar al estudio inicial
```

In [ ]:

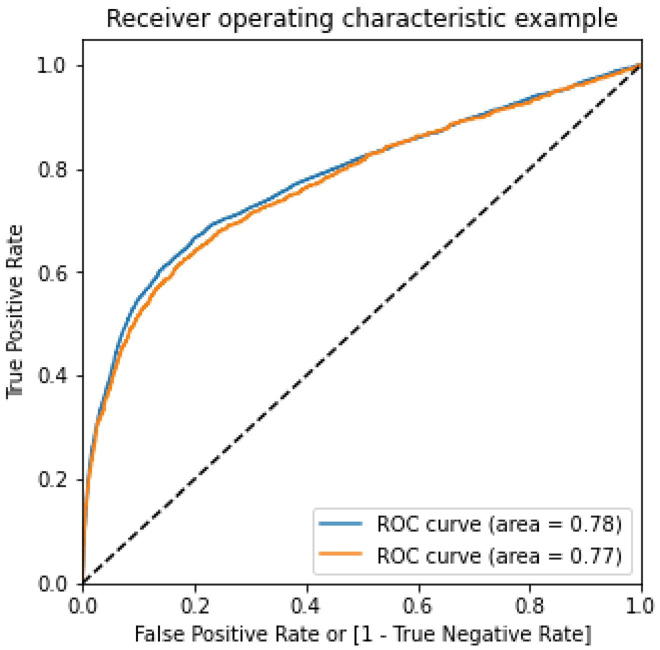
```
# Guardado del modelo de regresión logística
# 1. Se importa librería para guardado
import joblib
```

```
# 2. Mediante el método dump del paquete joblib se guarda el modelo con
joblib.dump(logreg, 'model_lm.pkl') # Guardo el modelo.
```

In [55]:

```
# Recuperacion del modelo y generación de la curva ROC
# 1. Carga del modelo guardado mediante función load de joblib
lm = joblib.load('model_lm.pkl') # Carga del modelo.
# 2. Predicción del modelo
y_train_pred_model_lm_grid, y_train_pred_model_lm_grid_prob, y_test_pred_

# 3. Se obtiene curva ROC de test y entrenamiento
draw_roc(y_train, y_train_pred_model_lm_grid_prob, y_test, y_test_pred_mo

# 4. Se imprime la precisión de acierto en entrenamiento y test
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_model_lm_g
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_model_lm_gri
```

```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator LogisticRegression from version 0.2
4.2 when using version 1.0.2. This might lead to breaking code or invalid
results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but LogisticRegression was fitted without f
eature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but LogisticRegression was fitted without f
eature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but LogisticRegression was fitted without f
eature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but LogisticRegression was fitted without f
eature names
  warnings.warn(
```




```
Accuracy train:  0.900780162506879
Accuracy test:  0.8998737496358162
```

6.2. Random Forest

Random Forest está formado por un conjunto de árboles de decisión, cada uno entrenado con una muestra ligeramente distinta de los datos de entrenamiento. Están basados en segmentar el espacio de los predictores en regiones simples para el manejo de las interacciones. Ventajas: . No requieren gran preprocesdo de datos (missing, one hot encoding ...) . No influeciado por los outliers . No es necesario preselección de variables . Gran potencia descriptiva con incorporación automática de interacciones . Equilibrio entre varianza y sesgo de un conjunto de árboles

Desventajas: . Intenta mejorar la poca fiabilidad . Complejo en la construcción.

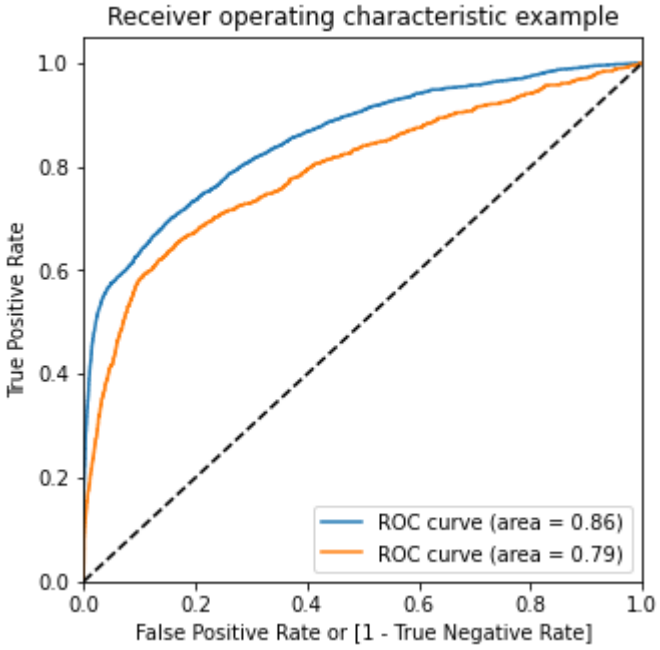
A continuación se implementa el ensamblado RandomForest del paquete sklearn. Se entrena el modelo con los siguientes hiperparámetros:

```
- n_estimators: Número de árboles a emplear
- max_depth: número máximo de nieveles en cada árbol de
decisión.
- max_features: número máximo de características para
dividir un nodo.
```

Para cada combinación de hiperparámetros, se utiliza validación cruzada repetida implementada en la función auxiliar 'training_model_hyperparameter' para su entrenamiento. Una vez entrenado, se obtiene la predicción para dibujar la curva ROC y calcular las precisión de acierto tanto en entrenamiento como en test.

In [56]:

```
# Entrenamiento y predicción del modelo RandomForest

# 1. Se importa paquete para ivocar el modelo RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

# 2. Creación de la malla de parámetros para random forest

param_grid_rf = [{'n_estimators': [30, 50, 75, 100],
                  'max_depth': [5, 6, 7, 8, 9, 10],
                  'max_features': [10, 15, 25, 30]}]
# 3. Se invoca el modelo
rf_model = RandomForestClassifier()
# 4. Entrenamiento que incluye hiperparámetros y validación cruzada repe
rf_grid = training_model_hyperparameter(rf_model, 'roc_auc', param_grid_

# 5. Predición del modelo obtenido
y_train_pred_rf, y_train_pred_rf_prob, y_test_pred_rf,y_test_pred_rf_prob

# 6. Representación de la curva ROC
draw_roc(y_train, y_train_pred_rf_prob, y_test, y_test_pred_rf_prob)

# 7. Se imprime el accuracy para test y entrenamiento
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_rf))
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_rf))
```




```
Accuracy train:  0.9181314946100806
Accuracy test:  0.8990968243177625
```

Análisis de Resultados

No mejora el accuracy con respecto al modelo lineal, obervando un ligero overfitting ya que se observa desplazamiento en la curva ROC de test. Se guarda el modelo en formato pkl. A continuación se guarda el modelo en formato pkl para luego descargarlo y comprobar que se obtienen resultados parecidos al informe inicial

In [ ]:

```
# Guardado del modelo de regresión logística

# 1. Se importa librería para guardado
import joblib
# 2. Mediante el método dump del paquete joblib se guarda el modelo con
joblib.dump(rf_grid, 'model_rf_grid.pkl') # Guardo el modelo.
```

In [57]:

```
# Carga del modelo Random Forest incocando a load de joblib
clf_rf_grid = joblib.load('model_rf_grid.pkl') # Carga del modelo.
```

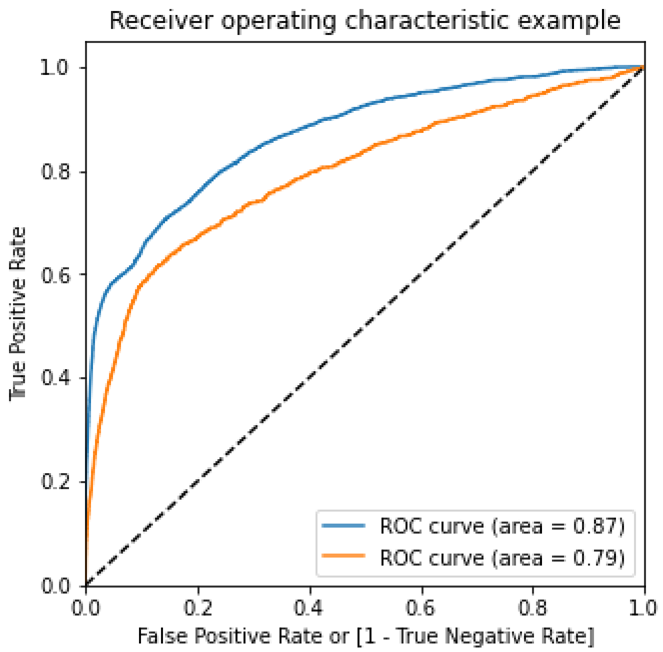
```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator DecisionTreeClassifier from version
0.24.2 when using version 1.0.2. This might lead to breaking code or inva
lid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator RandomForestClassifier from version
0.24.2 when using version 1.0.2. This might lead to breaking code or inva
lid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator GridSearchCV from version 0.24.2 wh
en using version 1.0.2. This might lead to breaking code or invalid resul
ts. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
```

```
aintainability-limitations
  warnings.warn(
```

In [58]:

```
# Generación del informe del modelo guardado Random Forest
# 1. Predicción del modelo
y_train_pred_rf_grid, y_train_pred_rf_grid_prob, y_test_pred_rf_grid,y_t

# 2. Representación del acurva ROC
draw_roc(y_train, y_train_pred_rf_grid_prob, y_test, y_test_pred_rf_grid_

# 3. Se imprime accuracy para test y train
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_rf_grid))
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_rf_grid))
```

```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but RandomForestClassifier was fitted witho
ut feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but RandomForestClassifier was fitted witho
ut feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but RandomForestClassifier was fitted witho
ut feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but RandomForestClassifier was fitted witho
ut feature names
  warnings.warn(
```



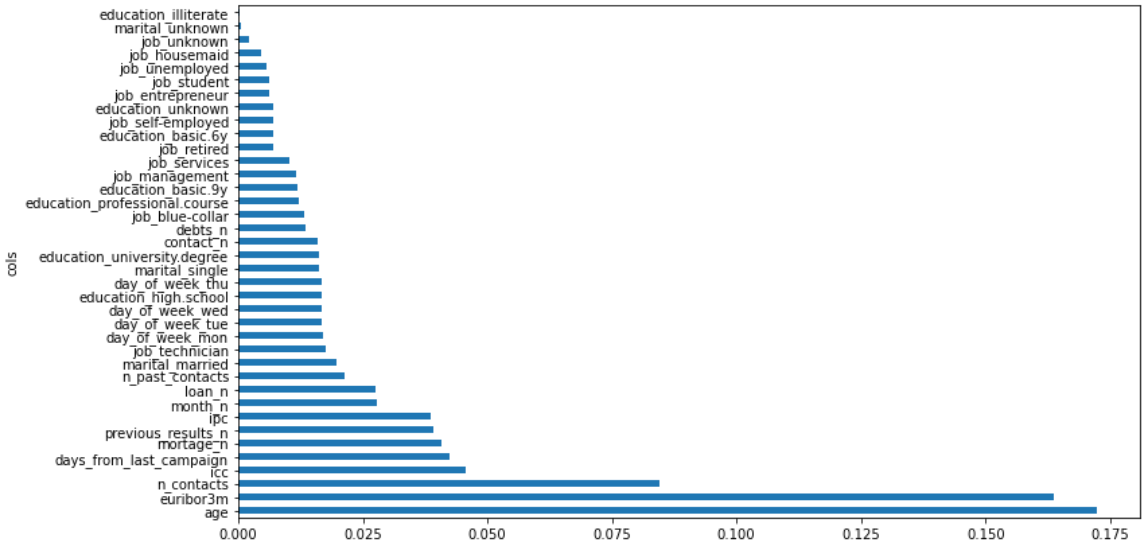

```
Accuracy train:  0.9209802207762778
Accuracy test:  0.9006506749538701
```

Importancia

El modelo Random Forest permite obtener un listado de importancia de las variables predictoras mediante el método feature*importances* . A continuación se obtiene el histograma ordenado por orden de importancia. Se observa que variables como edad e indicadores económicos como Euribor son más importantes que la variables del tipo de trabajo o educación.

In [63]:

```
# Importancia de Features co Random Forest
# 1. Entrenamiento o compilación del modelo
rf_model.fit(x_train, y_train)
# 2. Se llama a la función auxiliar para obtener orden
# de importancia en orden ascendente. Basado en feature_importances_
fi = rf_feat_importance(rf_model, x_test)
# 3. PLoteado del orden de importancia
plot_fi(fi[:40]);
```



6.3. Grading Boosting

Se basa en el ajuste de los modelos secuencialmente y la importancia (peso) de las observaciones va cambiando en cada iteración. Las predicciones van en la dirección de decrecimiento dada por el negativo del gradiente, de la función de error.

Ventajas:

- Facil de implementar y gran eficacia predictiva
- Contiene preprocesado automático (missing, variables categóricas ...) e importancia de vbles
- Posee buena escalabilidad

Desventajas:

- Dificil de interpretar.

Se puede observar que para su implementación se utiliza el modelo GradientBoostingClassifier del paquete sklearn.ensemble Para su entrenamiento se utiliza un grid de posibles parámetros, donde para cada combinación se utiliza la validación cruzada repetida y como criterio para su elección es el accuracy de la curva roc. Los parámetros a tunear son:

- n_iter_no_change: número de iteracciones consecutivas sin un mínimo grado de mejora tol
- n_estimators: número de árboles incluidos en el modelo
- learning_rate: reducción de la contribución de cada árbol.

Es importante que para que los resultados siempre sean reproducibles, existe la posibilidad de fijar el parámetro random_ssate a un valor fijo como se ha hecho en otras ocasiones.

El modelo ganador que se observa en la siguiente celda, contiene los siguientes parámetros: {'learning_rate': 0.15, 'n_estimators': 100, 'n_iter_no_change': 10}

In [64]:

```
# Implementación del modelo grading boosting
# 1. Se importa nuestra librería de funciones auxiliares
import import_ipynb
import functions
%run functions.ipynb
# 2. Se importa los paquete necesarios para entrenamiento del modelo
from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.model_selection import GridSearchCV

# 3. Se importa el paquete del modelo grading boosting
from sklearn.ensemble import GradientBoostingClassifier

# 4. Se importa el modelo obteniendo un objeto del tipo GradientBoosting
gb_model = GradientBoostingClassifier(verbose = 1)

# 5. Definición de la malla de parámetros para su tuneado
params_grid_gb = {
    "n_iter_no_change": ['None', 5, 10],
    "n_estimators": [30, 50, 75, 100],
    "learning_rate": [0.05, 0.1, 0.15]
}

# 6. Entrenamiento del modelo a través de una función auxiliar propia
gb_grid = training_model_hyperparameter(gb_model, 'roc_auc', params_grid
# 7. Visualización de los mejores parámetros obtenidos.
gb_grid.best_params_
```

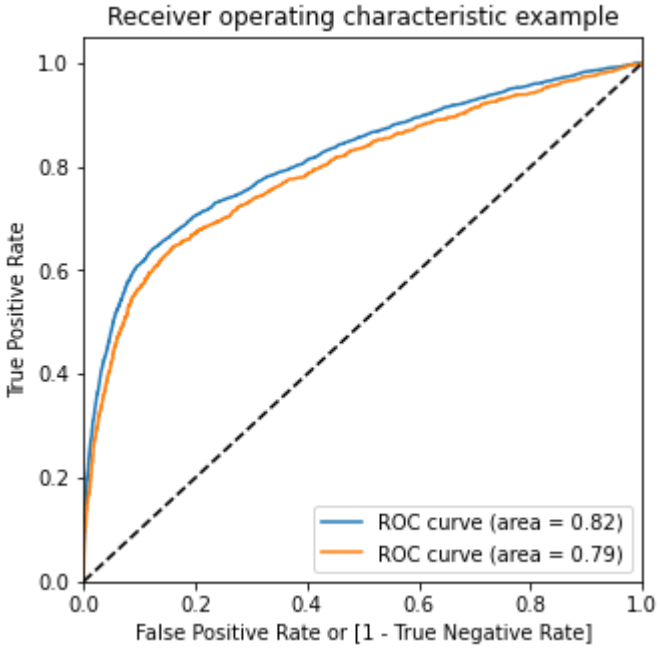
```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\model_selection\_
validation.py:372: FitFailedWarning:
120 fits failed out of a total of 360.
The score on these train-test partitions for these parameters will be set
to nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------
-------
120 fits failed with the following error:
Traceback (most recent call last):
  File "D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\model_sel
ection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\ensemble
\_gb.py", line 525, in fit
    self._check_params()
  File "D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\ensemble
\_gb.py", line 362, in _check_params
    raise ValueError(
ValueError: n_iter_no_change should either be None or an integer. 'None'
was passed

  warnings.warn(some_fits_failed_message, FitFailedWarning)
```

```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\model_selection\_
search.py:969: UserWarning: One or more of the test scores are non-finit
e: [       nan 0.78749127 0.78690129        nan 0.79091734 0.79079562
        nan 0.79820918 0.79815857        nan 0.79993897 0.79982978
        nan 0.7955774  0.79454158        nan 0.80038689 0.79975746
        nan 0.80229087 0.80147259        nan 0.80107599 0.80231603
        nan 0.80002404 0.79938391        nan 0.80158646 0.80205943
        nan 0.80140972 0.80184586        nan 0.79976337 0.80236962]
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\model_selection\_
search.py:969: UserWarning: One or more of the train scores are non-finit
e: [       nan 0.78963446 0.78934022        nan 0.79512077 0.7948059
        nan 0.80286481 0.80273085        nan 0.80574774 0.80572418
        nan 0.79885565 0.79829153        nan 0.80580056 0.80536444
        nan 0.80893085 0.80869184        nan 0.8082935  0.81122584
        nan 0.80488351 0.80454864        nan 0.80863468 0.81002775
        nan 0.80983407 0.81041335        nan 0.8074493  0.81219493]
  warnings.warn(
      Iter       Train Loss   Remaining Time
         1           0.6522           6.15s
         2           0.6259           6.10s
         3           0.6085           5.54s
         4           0.5967           5.59s
         5           0.5869           5.62s
         6           0.5798           5.61s
         7           0.5749           5.38s
         8           0.5704           5.37s
         9           0.5667           5.35s
        10           0.5644           5.26s
        20           0.5497           4.66s
        30           0.5416           4.05s
        40           0.5374           3.47s
        50           0.5340           2.91s
        60           0.5316           2.33s
        70           0.5283           1.75s
        80           0.5261           1.16s
        90           0.5241           0.58s
       100           0.5226           0.00s
```

Out[64]:
```
{'learning_rate': 0.15, 'n_estimators': 100, 'n_iter_no_change': 10}
```

Una vez compilado o entrenado el modelo, se procede a su predición y al trazado de su caurva ROC, así como el accuracy obtenido para su posterior comparación. Hasta ahora es el mejor resultado obtenido. En las posteriores celdas, se guarda dicho modelo para su posterior carga y comprobación de su predicción. Como a se realizó el entreamiento, no es necesario volver a realizarlo.

In [65]:
```
# Predicción y evaluación de resultados en Grading Boosting
from sklearn import metrics
# 1. Se importa para imprimir la gráfica de balanceo.
import matplotlib.pyplot as plt
# 2. Se impora el paquete para obtener el accuracy
from sklearn.metrics import accuracy_score
# 3. Predicción del modelo
y_train_pred_gb, y_train_pred_gb_prob, y_test_pred_gb, y_test_pred_gb_pr
                                    prediction_model(gb_grid, x_train, y
# 4. Gráfica ROC mediante función auxiliar propia
draw_roc(y_train, y_train_pred_gb_prob, y_test, y_test_pred_gb_prob)
# 5. Se imprime accuracy en train/test
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_gb))
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_gb))
```




```
Accuracy train:  0.9064452429510214
Accuracy test:  0.9002622122948432
```

In [ ]:

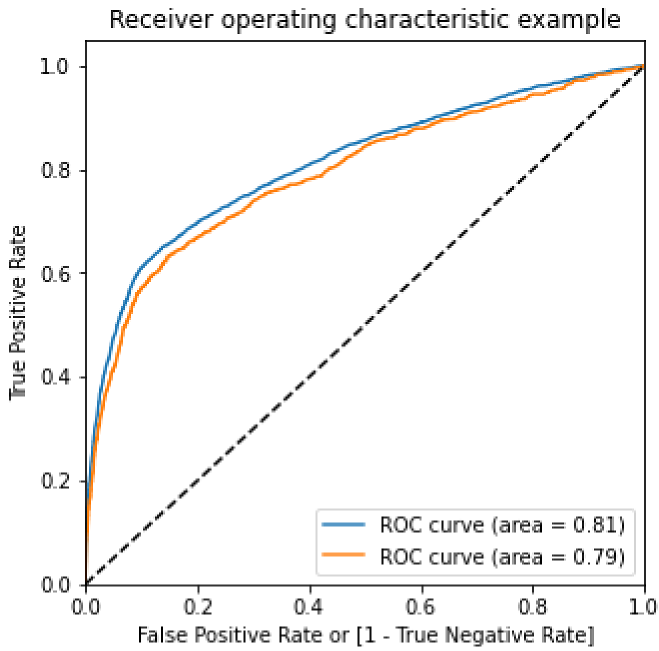
```
# Guardado del modelo en formato pkl mediante dump del paquete joblib
import joblib
joblib.dump(gb_grid, 'model_gb_grid.pkl') # Guardo el modelo.
```

In [66]:

```
# Recuperación del modelo fuardado y comprobación de su calidad
# 1. Descarga del modelo grading boosting guardado mediante método load
model_gb_grid = joblib.load('model_gb_grid.pkl') # Carga del modelo.
# 2. Predicción del modelo guardado
y_train_pred_model_gb_grid, y_train_pred_model_gb_grid_prob, y_test_pred

# 3. Curva ROC
draw_roc(y_train, y_train_pred_model_gb_grid_prob, y_test, y_test_pred_m

# 4. Accuracy en train y test del modelo guardado.
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_model_gb_
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_model_gb_gri
```

```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator GradientBoostingClassifier from ver
sion 0.24.2 when using version 1.0.2. This might lead to breaking code or
invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator DummyClassifier from version 0.24.2
when using version 1.0.2. This might lead to breaking code or invalid res
ults. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator DecisionTreeRegressor from version
0.24.2 when using version 1.0.2. This might lead to breaking code or inva
lid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
```

```
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator GridSearchCV from version 0.24.2 wh
en using version 1.0.2. This might lead to breaking code or invalid resul
ts. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but GradientBoostingClassifier was fitted w
ithout feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but GradientBoostingClassifier was fitted w
ithout feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but GradientBoostingClassifier was fitted w
ithout feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but GradientBoostingClassifier was fitted w
ithout feature names
  warnings.warn(
```




```
Accuracy train:  0.9046000453206435
Accuracy test:  0.9016218316014373
```

6.4. Support Vector Machine

Es un clasificador discriminativo definido formalmente por un hiperplano separador. Dicho separador puede ser lineal u a través de otro kernel o dimensión. Por tanto los parámetros del grid son:

- kernel: tipo de kernel usado. Puede ser lineal, polinomial (grado de polinomio), rbf gaussiano (parámetro gamma)
- degree: grado del kernel. En caso de que sea polinomial se puede utlizar de grado 2.

Ventajas:

- Pocos parámetros a tunear.

Inconvenientes:

- Lento en la optimización.
- El preprocesado es importante
- Resultado de calificación binaria sin reportar probabilidades

En el siguiente bloque se observa la implementacion de SVM mediantete la funcion SVC donde se le pasa el listado del kernel posible: polinomio grado 2, lineal o gaussiano. Al entrenarlo, se obtiene mejores resultados con el kennel gaussiano. El valor de C por defecto es uno. Su aumento podría significar menor sesgo y mayor sobreajuste. Se obtienen valores de accuracy por debajo que los anteriores. Quizás se podría mejorar con un mejor preprocesado de datos, pero de momento este modelo se descarta. Como con el resto de modelos, se guarda en formato pkl

In [67]:
```
# Modelling SVM
#  1. Se importa paquete SVC de svm
from sklearn.svm import SVC
# 2. Se instancial el modelo SVC
svc_model = SVC()
# 3. Se define la malla de hiperparámetros
params_grid_svc = {
    "kernel": ["linear", "rbf"],
    "degree": [1, 2]
}

# 4. Entrenamiento del modelo
svc_grid = training_model_hyperparameter(svc_model, 'roc_auc', params_gr
# 5. Visualizació de los mejores parámetros obtenidos
svc_grid.best_params_
```

Out[67]:
```
{'degree': 1, 'kernel': 'rbf'}
```

In [68]:
```
# Se imprime el accuracy tanto en test como train de svc
print("Accuracy train: ", accuracy_score(y_train, svc_grid.predict(x_tra
print("Accuracy test: ", accuracy_score(y_test, svc_grid.predict(x_test)
```

```
Accuracy train:  0.8995824026415461
Accuracy test:  0.8997766339710596
```

In [ ]:
```
# Se guarda el modelo en formato pkl
import joblib
joblib.dump(svc_grid, 'model_svm_grid.pkl') # Guardo el modelo.
```

In [69]:
```
# Recuperacion del modelo svc para evaluar el accuracy
# 1. De descarga el modelo guardado en formato pkl con load de joblib
model_svm_grid = joblib.load('model_svm_grid.pkl') # Carga del modelo.
#y_train_pred_model_svm_grid, y_train_pred_model_svm_grid_prob, y_test_p

# 2. IMpresion del accuracy en train/test para contrastarlo con el accur
print("Accuracy train: ", accuracy_score(y_train, model_svm_grid.predict
print("Accuracy test: ", accuracy_score(y_test, model_svm_grid.predict(x
```

```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator SVC from version 0.24.2 when using
version 1.0.2. This might lead to breaking code or invalid results. Use a
```

```
t your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator GridSearchCV from version 0.24.2 wh
en using version 1.0.2. This might lead to breaking code or invalid resul
ts. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but SVC was fitted without feature names
  warnings.warn(
Accuracy train:  0.8995824026415461
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but SVC was fitted without feature names
  warnings.warn(
Accuracy test:  0.8997766339710596
```

6.5. Xgboost

Está basado en modelos de grading boosting pero con la utilización de la regularización. Esta técnica está orientada a la reducción del sobreajuste y consiste fundamentalmente en una función de penalización para cada árbol costruído (número de hojas, pesos y predicción en cada hoja) Vantajas del modelo:

- Incorporan la regularización: pudiendo añadir mayor sesgo pero menos varianza
- Rápido
- Sorteo de vbles y observaciones

Desventajas del modelo:

- Hay que controlar la penalización: no siempre dabuenos resultados.

Para su implementación se debe instalar el paquete xgboost e invocar el modelo XGBClassifier. La malla de parámetros que se ha definido consiste en:

- max_depth: profundidad o número de nodos de bifurcación. A menor cantidad menor overfitting
- max_features: número máximo de características para dividir un nodo.

  Existe otros parámetros que podrían ser de interés:

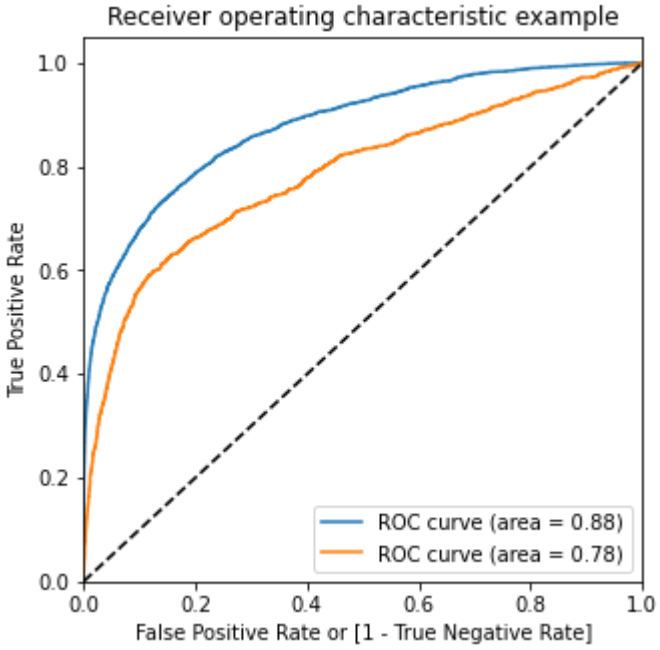
- eta: tasa de aprendizaje del modelo. Compromiso entre rapidez y optimización del modelo
- nrounds: número de iteraciones del modelo
- nthread: número de hilos computaciones para la ejecución en paralelo de varios procesos.

  Por último se entrena y se obtiene la curva ROC y el accuracy. Como en otras ocasiones se guarda y se comprueba que se ha recuperado obtieniendo resultados parecidos.

In [70]:

```
# Modelado XGBoost
# 1. Se importa las librerías necesarias
!pip install xgboost
from xgboost import XGBClassifier

# 2. Se invoca al modelo XGBClassifier
xgb_model = XGBClassifier()
# 3. Se definen los hiperparámetros
param_grid_xgb = {
                'max_depth': [5, 7],
                'max_features': [10, 20, 30]}

# 4. Entrenamiento del modelo con hiperprámetros
xgb_grid = training_model_hyperparameter(xgb_model, 'roc_auc', param_gri

# 5. Predicción del modelo
y_train_pred_xgb, y_train_pred_xgb_prob, y_test_pred_xgb, y_test_pred_xgl
                                        prediction_model(xgb_grid, x_train,

# 6. Curva ROC
draw_roc(y_train, y_train_pred_xgb_prob, y_test, y_test_pred_xgb_prob)

#7. Se imprime accuracy train y test
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_xgb))
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_xgb))
```

```
Requirement already satisfied: xgboost in d:\users\rmsalvador\anaconda3\l
ib\site-packages (1.5.2)
Requirement already satisfied: scipy in d:\users\rmsalvador\anaconda3\lib
\site-packages (from xgboost) (1.7.1)
Requirement already satisfied: numpy in d:\users\rmsalvador\anaconda3\lib
\site-packages (from xgboost) (1.20.3)
D:\Users\rmsalvador\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated and
will be removed in a future release. To remove this warning, do the follo
wing: 1) Pass option use_label_encoder=False when constructing XGBClassif
ier object; and 2) Encode your labels (y) as integers starting with 0, i.
e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[12:09:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_releas
e_1.5.1/src/learner.cc:576:
Parameters: { "max_features" } might not be used.

  This could be a false alarm, with some parameters getting used by langu
age bindings but
  then being mistakenly passed down to XGBoost core, or some parameter ac
tually being used
  but getting flagged wrongly here. Please open an issue if you find any
such cases.


[12:09:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_releas
e_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evalu
ation metric used with the objective 'binary:logistic' was changed from
'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
```




```
Accuracy train:  0.9235052280599527
Accuracy test:  0.8985141303292221
```

Se observa que tiene un accuracy en entrenamiento muy buena pero en test baja

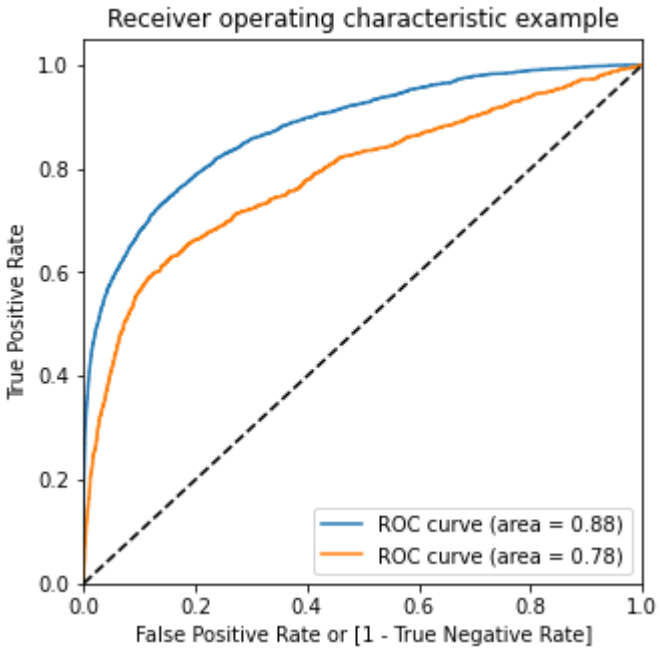
In [ ]:

```
# Guardado del modelo en formato pkl con dump del paquete joblib
import joblib
joblib.dump(xgb_grid, 'model_xgb_grid.pkl') # Guardo el modelo.
```

In [71]:

```
# Recuperacioon del modelo xgboost y evaluacion de su calidad de predicc
# 1. Recuperacion del modelo  mediante load del paquete joblib
model_xgb_grid = joblib.load('model_xgb_grid.pkl') # Carga del modelo.
# 2. Predicción del modelo xgboost
y_train_pred_model_xgb_grid, y_train_pred_model_xgb_grid_prob, y_test_pr

# 3. Curva ROC
draw_roc(y_train, y_train_pred_model_xgb_grid_prob, y_test, y_test_pred_

# 4. Accuracy en test y train
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_model_xgb
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_model_xgb_gr
```

```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator GridSearchCV from version 0.24.2 wh
en using version 1.0.2. This might lead to breaking code or invalid resul
ts. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
```




```
Accuracy train:  0.9235052280599527
Accuracy test:  0.8985141303292221
```

6.6. Arbol de Decision

Son modelos predictivos formados por reglas binarias que consiste en repartir las observciones en función de sus atributos y predecir así el valor de la variable respuesta. En este caso sólo está basado en un único árbol.

Los parámetros de para la generación del árbol son:

a. 'max_depth': profundidad máxima que puede alcanzar el árbol

b. 'min_samples_split': mínimo número de observaciones de un nodo para dividirse

c. 'min_samples_leaf': mínimo número de observaciones que debe de tener cada uno de los nodos hijos para que se produzca la división

d. 'criterion' : criterio utilizado para el cálculo de los puntos de corte.

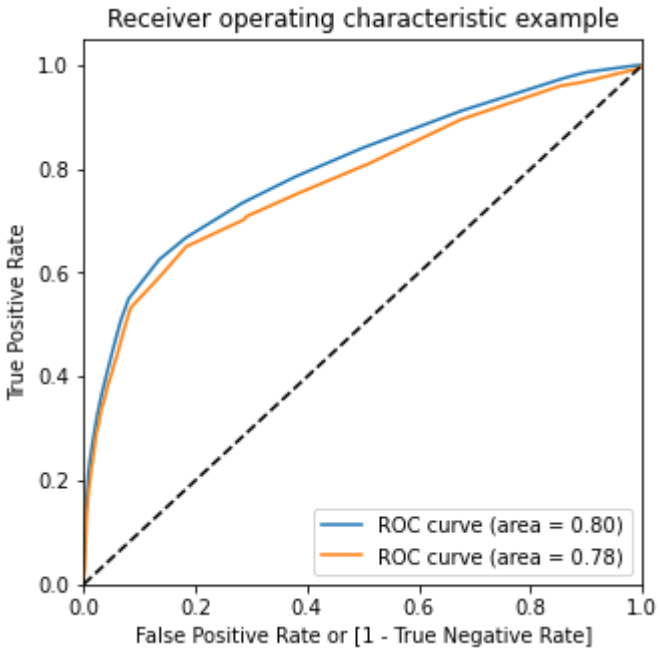
La implementación de este modelo se basa en el paquete DecissionTreeClassifier, donde se le pasa una malla con los hiperparámetros explicados anteriormente para su entrenamiento. Los mejores parámetros obtenidos son: {'criterion': 'gini', 'max_depth': 6, 'min_samples_leaf': 10, 'min_samples_split': 25}

A continuación se muestra los bloques de para evaluación del modelo y guardado.

In [72]:

```
# Modelado del modelo Arbol de Decision
# 1. Se importa las librerías necesarias
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree C
import numpy as np
# 2. Se invoca al modelo DecisionTreeClassifier
dtree = DecisionTreeClassifier(random_state=0)

# 3. malla de hiper-parámetros.
param_grid_dtree = {'max_depth':np.arange(5,15),
                    'min_samples_split': np.arange(5,30,1),
```

```
                    'min_samples_leaf':np.arange(5,30),
                   'criterion': ('gini', 'entropy')}

# 4. Entrenamiento
dtree_grid = training_model_hyperparameter(dtree, 'roc_auc', param_grid_

# 5. Predicción
y_train_pred_dtree, y_train_pred_dtree_prob, y_test_pred_dtree, y_test_p
                                        prediction_model(dtree_grid, x_train

# 6. Curva ROC
draw_roc(y_train, y_train_pred_dtree_prob, y_test, y_test_pred_dtree_pro

# 7. Se imprime accuraccy en train y test.
print("Accuracy train dtree: ", accuracy_score(y_train, y_train_pred_dtr
print("Accuracy test dtree: ", accuracy_score(y_test, y_test_pred_dtree)
```




```
Accuracy train dtree:  0.9039526075555987
Accuracy test dtree:  0.9003593279595998
```

In [73]:
```
# Visulalizacion de los mejores parámetros obtenidos
dtree_grid.best_params_
```

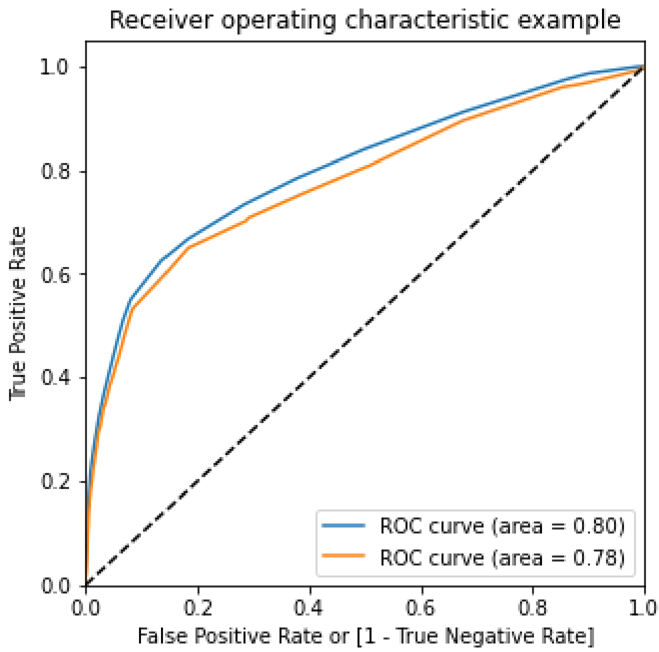
Out[73]:
```
{'criterion': 'gini',
 'max_depth': 6,
 'min_samples_leaf': 10,
 'min_samples_split': 25}
```

In [ ]:
```
# Guardado del modelo en formato pkl
import joblib
joblib.dump(dtree_grid, 'model_dtree_grid.pkl') # Guardo el modelo.
```

In [74]:
```
# Recuperación del modelo y contraste de calidad con el modelo origen.
# 1. Descarga del modelo mediante función load del paquete joblib
model_dtree_grid = joblib.load('model_dtree_grid.pkl') # Carga del model
# 2. Predicción del modelo árbol de decisión.
y_train_pred_model_dtree_grid, y_train_pred_model_dtree_grid_prob, y_tes

# 3. Curva ROC
draw_roc(y_train, y_train_pred_model_dtree_grid_prob, y_test, y_test_pre
```

```
# 4. Accuracy en train y test.
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_model_dtre
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_model_dtree_
```

```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator DecisionTreeClassifier from version
0.24.2 when using version 1.0.2. This might lead to breaking code or inva
lid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator GridSearchCV from version 0.24.2 wh
en using version 1.0.2. This might lead to breaking code or invalid resul
ts. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but DecisionTreeClassifier was fitted witho
ut feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but DecisionTreeClassifier was fitted witho
ut feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but DecisionTreeClassifier was fitted witho
ut feature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but DecisionTreeClassifier was fitted witho
ut feature names
  warnings.warn(
```



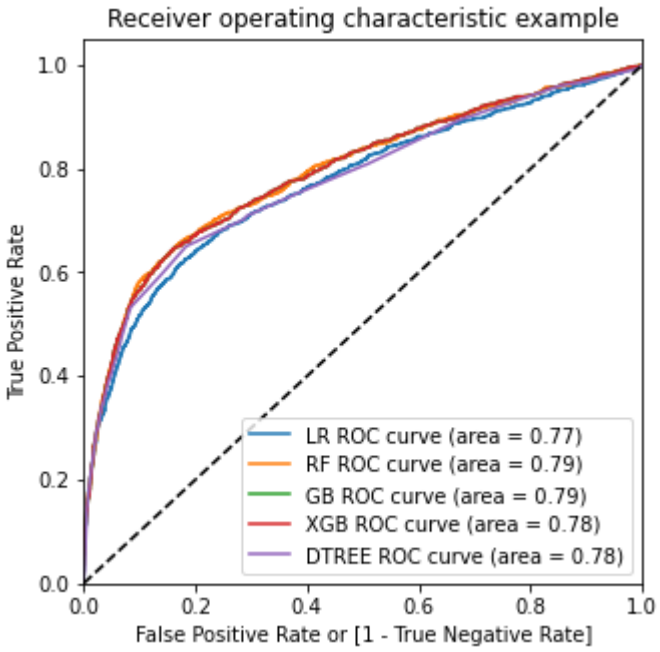

```
Accuracy train:  0.9039526075555987
Accuracy test:  0.9003593279595998
```

7. Comparativa de Modelos

Una vez analizado y estudiado los modelos más importantes así como explicado sus resultados con sus ventajas e inconvenientes, se procede a realizar una comparativa de los mismos. Para ello se obtiene una única gráfica con todas las curvas ROC de todos los modelos y el accuracy del test de cada uno. En el siguiente bloque se procede a realizar la comparativa. Se observa que se obtienen curvas ROC muy parecidas con accuracy ligeramente diferentes.

In [75]:

```
# Comparativa de modelos
# 1. Obtención de las métricas de la curva ROC para cada modelo
# y_test_pred_model_lm_grid_prob
# y_test_pred_lr_prob
lr_fpr, lr_tpr, lr_thresholds = metrics.roc_curve( y_test, y_test_pred_l
                                                   drop_intermediate = False

#y_test_pred_rf_grid_prob
#y_test_pred_rf_prob
rf_fpr, rf_tpr, rf_thresholds = metrics.roc_curve( y_test, y_test_pred_r
                                                   drop_intermediate = False

# y_test_pred_model_gb_grid_prob
# y_test_pred_gb_prob
gb_fpr, gb_tpr, gb_thresholds = metrics.roc_curve( y_test, y_test_pred_gl
                                                   drop_intermediate = False
# y_test_pred_model_xgb_grid_prob
# y_test_pred_xgb_prob
xgb_fpr, xgb_tpr, xgb_thresholds = metrics.roc_curve( y_test, y_test_pre
                                                   drop_intermediate = False

# y_test_pred_model_dtree_grid_prob
# y_test_pred_dtree_prob
dtree_fpr, dtree_tpr, dtree_thresholds = metrics.roc_curve( y_test, y_te
                                                   drop_intermediate = False

# 2. Obtencion del área de la curva ROC para cada modelo
lr_auc_score = metrics.roc_auc_score( y_test, y_test_pred_lr_prob )
rf_auc_score = metrics.roc_auc_score( y_test, y_test_pred_rf_prob )
gb_auc_score = metrics.roc_auc_score( y_test, y_test_pred_gb_prob )
xgb_auc_score = metrics.roc_auc_score( y_test, y_test_pred_xgb_prob )
dtree_auc_score = metrics.roc_auc_score( y_test, y_test_pred_dtree_prob

# 3. Dimensionado del plot
plt.figure(figsize=(5, 5))

# 4. Ploteado de las curvas y etiquetado con el área ROC de cada modelo
plt.plot( lr_fpr, lr_tpr, label='LR ROC curve (area = %0.2f)' % lr_auc_s
plt.plot( rf_fpr, rf_tpr, label='RF ROC curve (area = %0.2f)' % rf_auc_s
plt.plot( gb_fpr, gb_tpr, label='GB ROC curve (area = %0.2f)' % gb_auc_s
plt.plot( gb_fpr, gb_tpr, label='XGB ROC curve (area = %0.2f)' % xgb_auc_
plt.plot( dtree_fpr, dtree_tpr, label='DTREE ROC curve (area = %0.2f)' %

# 5. Etiqutado del plot con su ubicación
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
# 6. Se hace visible el plot
plt.show()
```



In [76]:

```
print("Accuracy test regression logistic: ", accuracy_score(y_test, y_te
print("Accuracy test random forest: ", accuracy_score(y_test, y_test_pre
print("Accuracy test grading boosting: ", accuracy_score(y_test, y_test_
print("Accuracy test svm : ", accuracy_score(y_test, svc_grid.predict(x_
print("Accuracy test x grading boosting: ", accuracy_score(y_test, y_tes
print("Accuracy test dtree: ", accuracy_score(y_test, y_test_pred_dtree)
```

```
Accuracy test regression logistic:  0.8998737496358162
Accuracy test random forest:  0.8990968243177625
Accuracy test grading boosting:  0.9002622122948432
Accuracy test svm :  0.8997766339710596
Accuracy test x grading boosting:  0.8985141303292221
Accuracy test dtree:  0.9003593279595998
```

Los mejores modelos en cuanto a predicción y AUC son grading boosting y arbol de decisión en cuanto a 'accuracy' y AUC. Es importante señalar que se evaluaron los modelos por segunda vez y se guardó la primera iteración de cada uno de los modelos en formato pkl. La comparativa mostrada se corresponde con la segunda iteración. Sin embargo en la primera iteración obtuvo mejor resultado grading boosting que árbol de decisión. De hecho se es el mejor resultado obtenido hasta ahora: 'Accuracy test' ---> 0.9016218316014373 Aunque en ambas iteraciones fueron los modelos ganadores.

Para evitar aleatoriedad, se aconseja de cara a futuro, fijar las semillas

Como se observa en los anteriores bloques, los resultados de la primera iteración (formatos en disco pkl) son muy parecidos a los de la segunda.

Por tanto se escoge de momento el modelo guardado en disco para utilizarlo en producción Flask que se verá a continuación: 'model_gb_grid'

8. Otros modelos analizados.

Aparte de los modelos más representativos, se han analizado otros modelos para comprobar si mejoran con respecto a los ya analizados. Se trata de bagging, red neuronal y knn.

8.1. Bagging

Consiste en la generación de varios árboles ponderando el resultado de cada uno de ellos. Para su implementación en python, se utiliza el paquete BaggingClassifier, donse se incluye los siguientes parámetros:

- base_estimator: Estimador basado en árbol de decisión.
- n_estimators: número de árboles o estimadores a usar. En este caso 500

A su vez los parámetros del árbol de decisión son:

- min_samples_split: al menos debe tener 5 observaciones para poder dividir el nodo.
- min_samples_leaf: al menos 25 observaciones debe tener cada uno de los nodos hijos para que se produzca la división
- max_depth: profundidad máxima de 8 que puede alcanzar el árbol.
- criterion: criterio gini para calcular los puntos de corte.

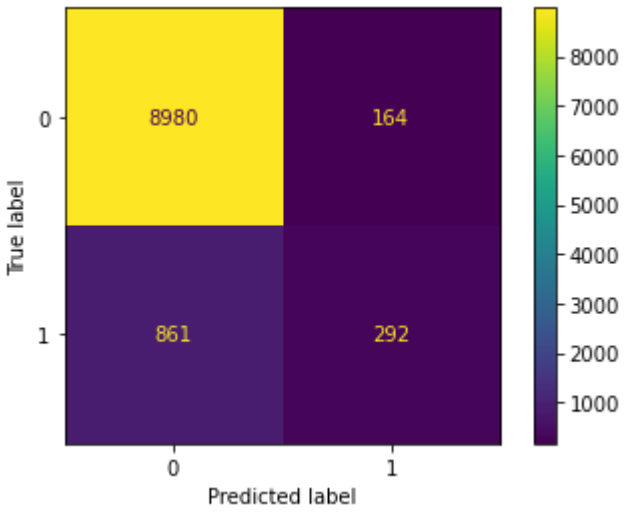
En los siguientes bloques se puede observar el entrenamiento del modelo bagging, asi como su predicción, matriz de confución, curva ROC y accuracy. Se puede observar que ha dado muy buenos resultados. Por lo tanto, dicho modelo es uno de los candidatos a tner en cuenta. Se guarda por si se quiere explotar en un futuro. La matriz de confusión, así como la tabla de clasificación, se interpreta de la misma manera que el modelo lineal. El parámetro f1-score es un fórmula basada en los parámetros 'recall' y 'precision' Simplemente comentar que el nombre del modelo aparece etiquetado como 'Logistic Regresion' porque se puso la variable name correspondiente al modelo anterior. La próxima iteracción ya se ha cambiado a la etiqueta name_bagging, con lo cual debería aparecer con el nombre 'Bagging'.

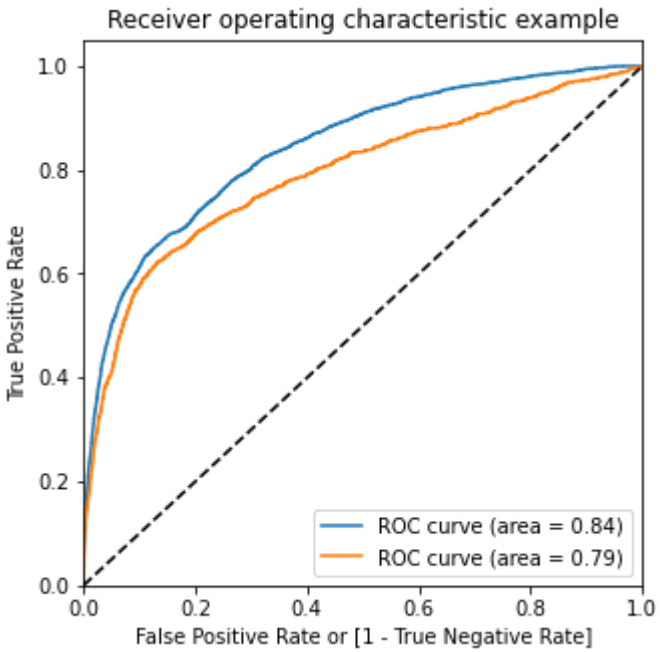
In [77]:
```
# bagging classifier
# 1. Se importa la libería  BaggingClassifier de paquete ensemble de skl
from sklearn.ensemble import BaggingClassifier
# 2. Se instancia el modelo con los parámetros comentados anteriormente.
model_bagging = BaggingClassifier(base_estimator = DecisionTreeClassifie
```

In [78]:
```
# 3 Se define la validación cruzada repetida con 10 splits y en modo ale
# Se fija la semilla
kfold = model_selection.KFold(n_splits=10, shuffle = True, random_state=
# 4. Se lanza la validación cruzada repetida para su entrenammiento
cv_results_bagging = model_selection.cross_val_score(model_bagging, x_tr
# 5. Se imprime los resultados con la media y desviación.
name_bagging='Bagging'
msg = "%s: %f (%f)" % (name_bagging, cv_results_bagging.mean(), cv_resul
print(msg)
model_bagging.fit(x_train, y_train)

# 6. Se obtiene el informe de clasificación  medainte función auxiliar p
report(model_bagging)

# 7. Predicción del modelo
y_train_pred_bagging, y_train_pred_bagging_prob, y_test_pred_bagging,y_t

# 8. Grafica ROC
```

```
draw_roc(y_train, y_train_pred_bagging_prob, y_test, y_test_pred_bagging_

# 9. Se imprime accuracy de test y entrenamiento.
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_bagging))
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_bagging))
```

```
Logistic Regression: 0.901039 (0.003863)
              precision    recall  f1-score   support

           0       0.98      0.91      0.95      9841
           1       0.25      0.64      0.36       456

    accuracy                           0.90     10297
   macro avg       0.62      0.78      0.65     10297
weighted avg       0.95      0.90      0.92     10297
```






```
Accuracy train:  0.9053769706386974
Accuracy test:  0.9004564436243566
```

In [ ]:

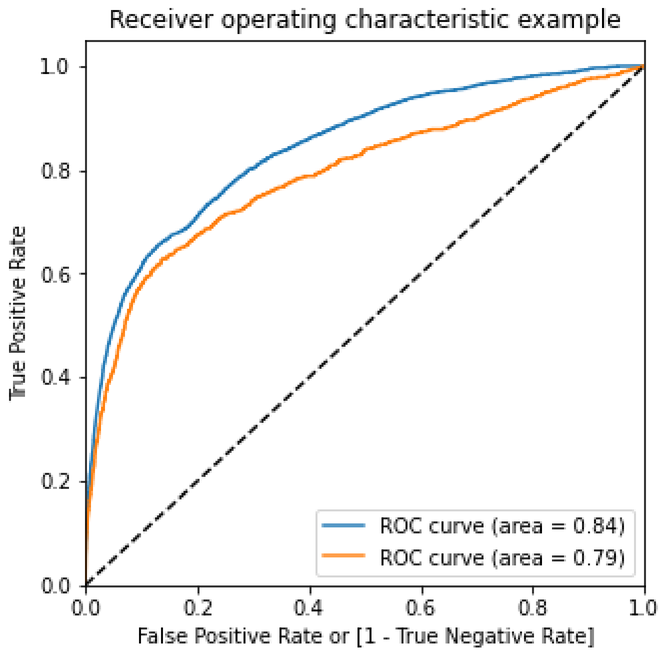
```
# Guardado del modelo en formato pkl
import joblib
joblib.dump(model_bagging, 'model_bagging.pkl') # Guardo el modelo.
```

In [79]:

```
# Recuperación del modelo y evaluación de calidad
```

```
# 1. Desrga del modelo guardado medainte load de la libería joblib
bagging = joblib.load('model_bagging.pkl') # Carga del modelo.
# 2. Predicción del modelo
y_train_pred_model_bagging_grid, y_train_pred_model_bagging_grid_prob, y_

# 3. Curva ROC
draw_roc(y_train, y_train_pred_model_bagging_grid_prob, y_test, y_test_p

# 4. Se imprime accuracy train/test
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_model_bag
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_model_baggin
```

```
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator DecisionTreeClassifier from version
0.24.2 when using version 1.0.2. This might lead to breaking code or inva
lid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:329: User
Warning: Trying to unpickle estimator BaggingClassifier from version 0.2
4.2 when using version 1.0.2. This might lead to breaking code or invalid
results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-m
aintainability-limitations
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but BaggingClassifier was fitted without fe
ature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but BaggingClassifier was fitted without fe
ature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but BaggingClassifier was fitted without fe
ature names
  warnings.warn(
D:\Users\rmsalvador\Anaconda3\lib\site-packages\sklearn\base.py:443: User
Warning: X has feature names, but BaggingClassifier was fitted without fe
ature names
  warnings.warn(
```



```
Accuracy train:  0.9050208798679227
Accuracy test:  0.9008449062833835
```

8.2. Red Neuronal

Dicho modelo está compuesto por una serie de capas donde se define el número de neuronas y su función de activación en cada capa.Cada neurona está compuesta por una suma ponderada de todos los valores de entrada más un valor bias. Por tanto la matriz de pesos para esa capa está compuesta de N filas/neuronas de la capa anterior x M neuronas/columnas de la capa actual.
Las funciones de activación más típicas suele ser sigmoide donde da como resultado 0 o 1 en función del valor de entrada o una relu donde realiza un filtrado de los valores negativos.

En el siguiente bloque se define una red neuronal compuesta por:

- Una capa de entrada de 38 entradas , ya que se compone de 38 variables en el procesado del dataset.
- Una capa de normalización de datos para obtener mejores resultados.
- Dos capas ocultas, una de 64 neuronas u otra de 16
- Una capa de drop out para mejorar el rendimiento
- Por ultima la capa de salida que predicirá la clase.

Su arquitectura de red se define mediante el api de keras. Y las funciones de activación en capas intermedias suele ser Relu y en la capa final sigmoide.

Para su entrenamiento y optimización, al ser un problema de clasificación binaria, se utiliza como función de pérdida la logarítimca 'binary_crossentropy' y el algoritmo de descenso de gradiente eficiente 'adams'. Y para evaluar su calidad, se utiliza como métrica la exactitud o accuracy.

La malla de parámetros definida es:

- epochs: es el número de ves que se ejecuta el algoritmo
- batch_size: es el número de datos que tiene cada iteración de un cliclo (epoch)

En el el siguiente bloque se puede observar los detalles de la implementación.

Se concluye que el 'accuracy' no mejora con respecto a grading boosting o arbol de decisión que son los que mejor resultado hasta ahora han dado.

In [80]:

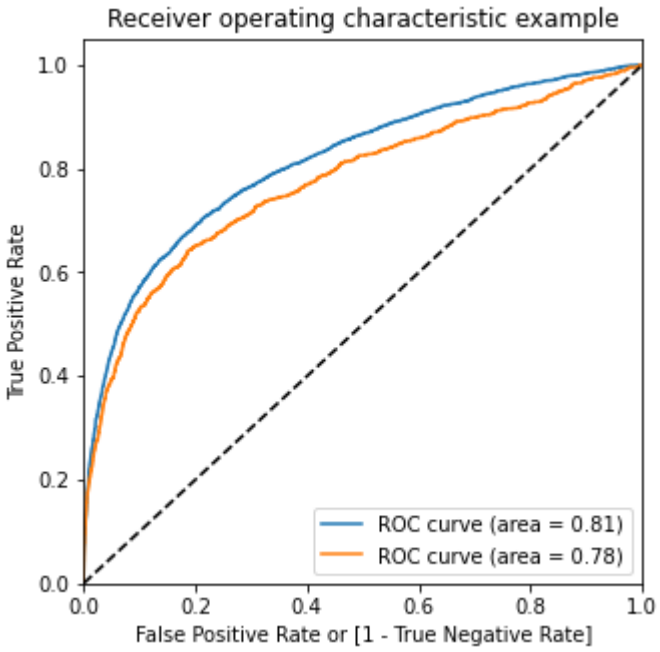
```
# Implementacion de red neuronal
# Para las redes neuronales se utiliza el paquete keras & TensorFlow
# 1. Se importan las librerías necesarias: keras, Sequential, ...
#!pip install --upgrade tensorflow

from tensorflow import keras
import tensorflow as tf
from keras.models import Sequential
from keras.wrappers.scikit_learn import KerasClassifier
from keras.layers.core import Dense, Dropout, Activation
#x_train,x_test,y_train,y_test = train_test_split(x,y, random_state=42)

# 2. Definición de la arquitectura de lka red neuronal
```

```
def create_model():
  model = keras.Sequential([
    keras.layers.Flatten(input_shape=(38,)),
    tf.keras.layers.BatchNormalization(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(1, activation='sigmoid')
  ])
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['
  return model

# 3. Se invoca la red neuronal
model_nn = KerasClassifier(build_fn=create_model, verbose=0)

# 4. Se define la malla de hiper-parámetros.
batch_size = [100, 150]
epochs = [10]
params_grid_nn = dict(batch_size=batch_size, epochs=epochs)

# 5. Entrenamiento del modelo con la malla de parámetros
nn_grid = training_model_hyperparameter(model_nn, 'roc_auc', params_grid
nn_grid.best_params_
#grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1,

# 6. Se imprime los mejores parámetros obtenidos
print("Best: %f using %s" % (nn_grid.best_score_, nn_grid.best_params_))
```

```
C:\Users\RMSALV~1\AppData\Local\Temp/ipykernel_13640/901755133.py:25: Dep
recationWarning: KerasClassifier is deprecated, use Sci-Keras (https://gi
thub.com/adriangb/scikeras) instead.
  model_nn = KerasClassifier(build_fn=create_model, verbose=0)
Best: 0.782816 using {'batch_size': 150, 'epochs': 10}
```

In [81]:

```
# Predicción y cálculo del accuracy del modelo neuronal
# 1. Predicción del modelo
y_train_pred_nn, y_train_pred_nn_prob, y_test_pred_nn, y_test_pred_nn_pr
                                        prediction_model(nn_grid, x_train, y
# 2. Curva ROC
draw_roc(y_train, y_train_pred_nn_prob, y_test, y_test_pred_nn_prob)
# 3. Se imprime accuracy train/test
print("Accuracy train: ", accuracy_score(y_train, y_train_pred_nn))
print("Accuracy test: ", accuracy_score(y_test, y_test_pred_nn))
```




```
Accuracy train:  0.9029490790197793
Accuracy test:  0.8996795183063028
```

8.3 Algoritmo Knn

Knn es un algoritmo basado en instancia supervisado donde busca las observaciones más cercanas para intentar predecir. Ventajas:

- Sencillo de implementar Inconveniente:
- Consume muchos recursos: memoria porque utiliza todo el dataset y procesamiento de CPU.

  Se basa en estos principios:

  1. Calcula la distancia del nuevo item con respecto al dataset de entrenamiento
  2. Selecciona los k elementos más cercanos o con menor distancia
  3. realiza una votación de la mayoría.

  Para desempate es importante elegir un valor impar de k y un valor k alto no implica mejor respuesta. Y para el cálculo de la distancia, se suele emplear la distancia Euclidiana.

A continuación, se obtiene el accuracy para un rago (1-40) de valores k. El mínimo error se obtiene para K=33 con un accuracy de 0.8988. Dicho valor no mejora el accuracy de los modelos basados en árbol.

In [82]:

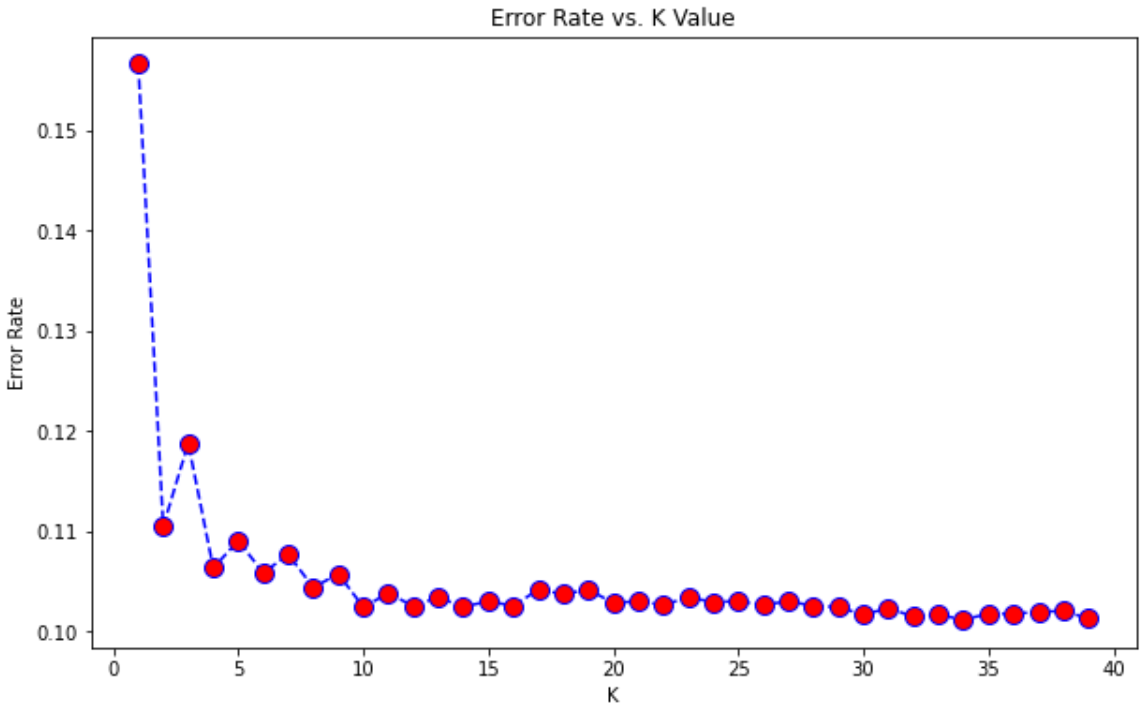
```
# Algoritmo Knn K-nearest neighbor
# Calcula las distancias teniendo en cuenta las k features más cercanas.

# 1. Se importa la librería propia auxiliar par el preprocesado de datos
import import_ipynb
import functions

# 2. Se imprime librería split train, test
# y otras: librería del modelo KNeighborsClassifier, numpy y metrics
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
```

```
from sklearn import metrics

# 3. Preprocesado del dataset
rename=True
df2 =transform(df_ref,rename)
y = df2['deposit_n']
x = df2.drop('deposit_n', axis = 1)

# 4. Split del conjunto de datos en train/test
x_train,x_test,y_train,y_test = train_test_split(x,y, random_state=42)

# 5. Para cada valor de k  del rango, se invoca al modelo para entrenarl
# Se obtiene un array con los valores de error rate y accuracy.
error_rate = []
acc = []
for i in range(1,40):
     knn = KNeighborsClassifier(n_neighbors=i)
     knn.fit(x_train, y_train)
     pred_i = knn.predict(x_test)
     error_rate.append(np.mean(pred_i != y_test))
     acc.append(metrics.accuracy_score(y_test, pred_i))
```

In [83]:

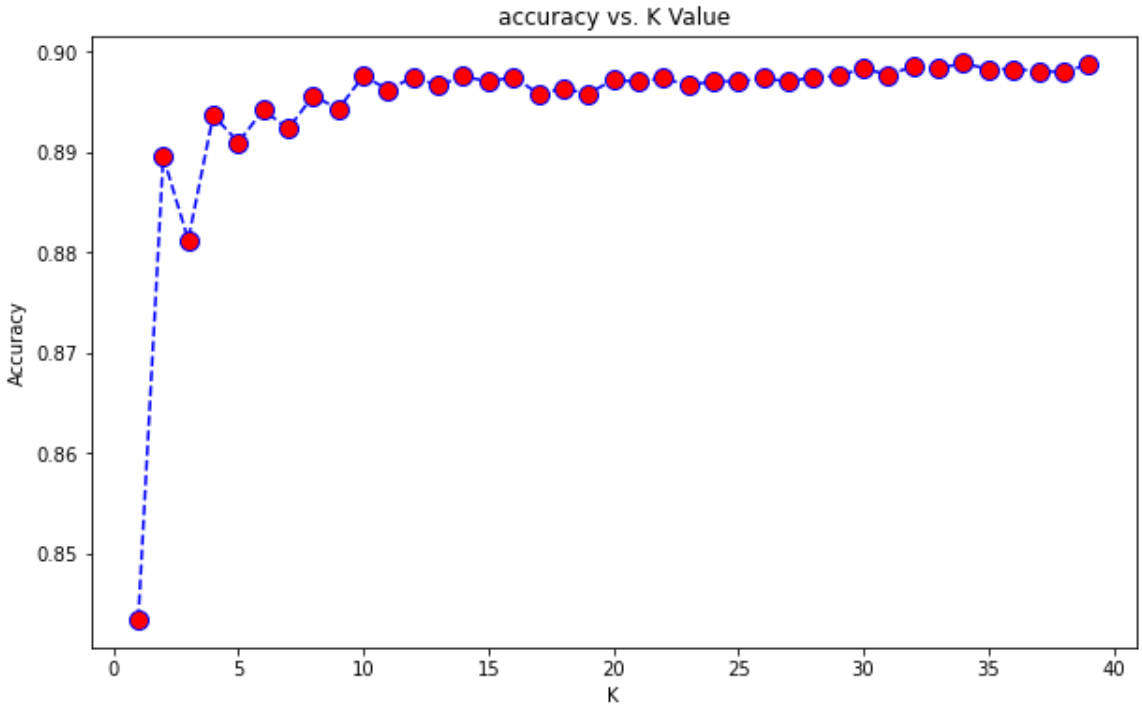
```
# Visualizacion del tuneado de parámetros en knn model

# 1. Se importa la librería pyplot par vissualizar el tuneado de parámet
import matplotlib.pyplot as plt
# 2. Dimensión del ploteado.
plt.figure(figsize=(10,6))
# 3. Se plotear el array Error rate y etiquetado.
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed',
         marker='o',markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
# 4. Se imprime el mínimo Error rate y la k correspondiente.
print("Minimum error:-",min(error_rate),"at K =",error_rate.index(min(er
```

```
Minimum error:- 0.10119452267650772 at K = 33
```



In [95]:
```
# Se visualiza el accuracy
# 1. Dimensión del ploteado
plt.figure(figsize=(10,6))
# 2. PLoteado del array accuracy en función de k y etiquetado
plt.plot(range(1,40),acc,color = 'blue',linestyle='dashed',
         marker='o',markerfacecolor='red', markersize=10)
plt.title('accuracy vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy')
s=pd.Series(acc)
# 3. Se imprime el máximo accuracy y la k correspondiente.
print("Maximum accuracy:-",s.max(),"at K =",acc.index(s.max()))
```

```
Maximum accuracy:- 0.8988054773234923 at K = 33
```



9. Tecnicas Oversampling

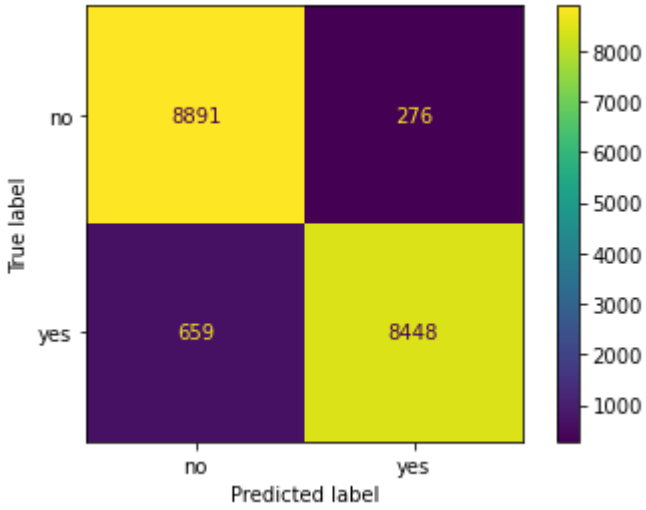
Técnica Over-sampling basado en los k-nearest vecinos para la creación de más casos de la clase minorista. Se observa que se obtiene un mayor accuracy de 0.95 con respecto a 0.9 para el modelo de regresion logística. Puede ser una alternativa a consisderar de cara a futuro pero de momento se descarta porque es una alteración de los datos de entrada, ya que se pasa de 41188 datos a 73096.

A continuación, se observa su implementación mediante la utilidad SMOTE de paquete imblearn.over_sampling El resto es la misma implementación que se hizo para regresión logística con su mmisma interpretación de resultados obtenidos.

In [96]:
```
# 1. Se importa nuestra librería de funciones auxiliares
import import_ipynb
import functions
%run functions.ipynb

# 2. Dependencia para el split train,test
from sklearn.model_selection import train_test_split

# 3. Librería panda para lectura del dataframe
import pandas as pd
```

```
# 4. Librerías para normalizacion de datos y métricas en general
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

# 5. Librería para el Modelo de regresion logística
# Así como dependencias para obtención de informe de clasificación
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix,Confu

# 6.Librería para over-sampling
from imblearn.over_sampling import SMOTE
#from imblearn.combine import SMOTETomek

# 7. Lectura del fataframe
url='bank-additional-full.csv'
df_ref = pd.read_csv(url,sep=";")

# 8. Procesado básico del dataframe
x=df_ref.drop(['y'],axis=1)
x=pd.get_dummies(x)
y=df_ref['y']


# 9. Instanciación del objeto Smote
smote = SMOTE()
# Otra alternativa de over-sampling mediante librería SMOTETomek
#smt = SMOTETomek()
#X_smt, y_smt = smt.fit_resample(x_train, y_train)

# 10. Obtencion de mas muestras para el equilibrado
print('Tamaño original ', len(x))
x, y = smote.fit_resample(x, y)
print('Tamaño remuestreado', len(x))

# 11. Split en test y train
x_train,x_test,y_train,y_test = train_test_split(x,y, random_state=42)


# 12. Escalado de los datos
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)

# 13. Modelling como regresión logística

logreg = LogisticRegression(solver='liblinear', random_state = 100)
logreg = logreg.fit(x_train, y_train)

# 14. Obtención del informe de clasificación
report(logreg)
```

```
Tamaño original  41188
Tamaño remuestreado 73096
              precision    recall  f1-score   support

          no       0.97      0.93      0.95      9550
         yes       0.93      0.97      0.95      8724

    accuracy                           0.95     18274
   macro avg       0.95      0.95      0.95     18274
weighted avg       0.95      0.95      0.95     18274
```



Análisis de Métricas

El conjunto de test o validación corresponde en esta ocasión con 9569 casos pertenecientes al no y 8705 al sí. Por tanto ahora está bastante equilibrado. A continuación se observa la precisión con la que se acertó. Informe de clasificación:

Precision: 97% de tasa de acierto para la no contratacion frente a un 93% de contratacion de deposito.

Recall o ratio verdaderos positivos o sensibilidad: 93 % para el no y 97% para el si.

Además se obtiene la matriz de consfusión con el detalle de los casos postitivos/negativos acertados y los falsos positivos/negativos.

10. Produccion Flask

Para poner en producción el modelo, se debe cargar el modelo seleccionado que se grabó con extensión pkl. Mediante la libería Flask se genera un servidor que está a la escucha en una IP y puerto. En este caso localhost y puerto 5000. Cuando se recibe una petición o solicitud de depósito de un nuevo cliente, se consumen los recursos a través de las API´s implementadas. Dichos recursos se definen con la palabra clave @app.route, en el cual se identifica el recurso y los métodos a implementar. En nuestro caso, se ha definido dos métodos GET y POST donde se rellena un formulario web y se predice sobre los datos introducidos, mandando una predicción en formato json donde se dice si se le concede el depóstino o no.

Para la recogida de los datos del formulario se ha empleado código html (form1.html) donde se define la página web con todos los campos a recoger y sus tipos. Además se le asocia la guia de estilo style_form1.css donde se diseña el aspecto físico de la propia página web. Además se le asocia el javascript form1.js para que el nombre descriptivo de cada campo del formulario se sitúe arriba cuando haya contenido o el foco esté en el propio formulario.

Por último para darle una acceso externo al servicio web local implementado, se lanza el ejecutable ngrok de tal manera que mediante un URL externa redirige todo su tráfico a la

URL local. Para ello hay que ejecutar desde el cmd de windows el siguiente comando: ngrok.exe http 5000.

In [ ]:

```
# Servicio que realiza predicción sobre una petición
# mediante formulario
# 1. Se importa nuestra librería de funciones auxiliares
import import_ipynb
import functions
%run functions.ipynb

# 2. Se importan todas las librerías necesarias para implementar dicho s
from flask import Flask, request, jsonify, render_template
import joblib # Se utiliza joblib para leer el modelo pre-entrenado
import pandas as pd

# Ademas se carga en memoria el dataframe original.
# Es importante que se realimente para su actualización.
# Por último se visualiza para comprobar su correcta carga en memoria

url='bank-additional-full.csv'
df_ref = pd.read_csv(url,sep=";")

# 3. Se crea la instancia de Flask
app = Flask(__name__)

# 4. Se abre el archivo que contiene el modelo candidato grading boostin
#MODEL_BANK_TEST = joblib.load('new_model_try2.pkl')
MODEL_BANK_TEST = joblib.load('model_gb_grid.pkl')


# 5. Se crean las etiquetas con las cuales se clasificaran nuevos datos
# Sabemos que se corresponde con la denegación o concesión del depósito.

MODEL_LABELS_BANK =['no','yes']
"""
El método predict sera el encargado de clasificar y dar una respuesta
a cualquier IP que le envie una petición.
"""
# 6. Recurso que sólo captura los datos del formulario.
@app.route('/form1')
def form1():
    return render_template('form1.html')

# 7. Recurso que genera una predicción mediante la recogida de datos de
# a través de un formulario.
@app.route('/form1+', methods=['POST','GET'])
def form1_():
    if request.method == "GET":
        return render_template('form1.html')
    # 8. Obtiene los datos del formulario en modo diccionario mediante e
    elif request.method== 'POST':
        data = request.form.to_dict()
        # 9. Se visualiza el diccionario de entrada. Se activa en modo d
        #print(data)
        #print(data.keys())
        #print(data.values())

        # 10. Conversión de datos adecuada de las variables numéricas
        data['age']= int(data['age'])
        data['campaign'] = int(data['campaign'])
        data['pdays'] = int(data['pdays'])
        data['previous'] = int(data['previous'])
```

```
        data['emp.var.rate'] = float(data['emp.var.rate'])
        data['cons.price.idx']= float(data['cons.price.idx'])
        data['cons.conf.idx']= float(data['cons.conf.idx'])
        data['euribor3m'] = float(data['euribor3m'])
        data['nr.employed'] = float(data['nr.employed'])

        # 11. Se formatea adecuadamente el nuevo registro para añadirlo
        data2={'contact': data['contact'], 'month': data['month'], 'day_
        'campaign': data['campaign'], 'pdays': data['pdays'], 'previous'
        'age': data['age'], 'job': data['job'],'marital': data['marital'
        'default': data['default'], 'housing': data['housing'],'loan': da
        'emp.var.rate': data['emp.var.rate'], 'cons.price.idx': data['co
        'cons.conf.idx':data['cons.conf.idx'] ,'euribor3m': data['euribo

        # 12. Se añade nuevo dato al dataframe original
        df1= df_ref.append(data2,ignore_index=True)

        # 13. Se procesa adecuadamente para alimentar al modelo ya entre
        rename=True
        df2 =transform(df1,rename)
        # 14. Se obtiene el último registro correspondiente del formular
        df2 = df2.drop('deposit_n', axis = 1)
        x1=df2.tail(1)

        # 15. Se visualiza el resgistro procesado junto con sus columnas
        print(x1)
        x1.columns

        # 16. Se realiza dicha predicción.
        label_index = MODEL_BANK_TEST.predict(x1)

        """
        La variable label contendra el resultado de la clasificación.
        """
        # 17. Se obtiene la etiqueta de dicha predicción
        label = MODEL_LABELS_BANK[label_index[0]]

        # 18. Se crea y se envía la respuesta al cliente
        return jsonify(status='Predicción Completada', prediccion=label)
        #return "Formulario recibido"
    else:
        return "Método no aceptado"

# 19. Recurso de página no encontrada
@app.errorhandler(404)
def page_not_found(error):
    return render_template('page_not_found.html'),404

# 20. Se inicia y se pone a la escucha el servidor.
if __name__ == '__main__':

    app.run(debug=False)
```

11. Otras predicciones

El umbral por defecto está en 0.5 de tal manera que para probabilidades mayores del umbral por defecto lo consiera una concesión de depóstio y por debajo una denegación. Se ha cambiado el umbral a 0.3 de tal manera que se quiere detectar más casos positivos, existiendo por tanto más casos de falsos positivos y menores falsos negativos.

La tasa de acierto o precision para casos positivos lógicamente ha mejorado de un 22% a un 36%.

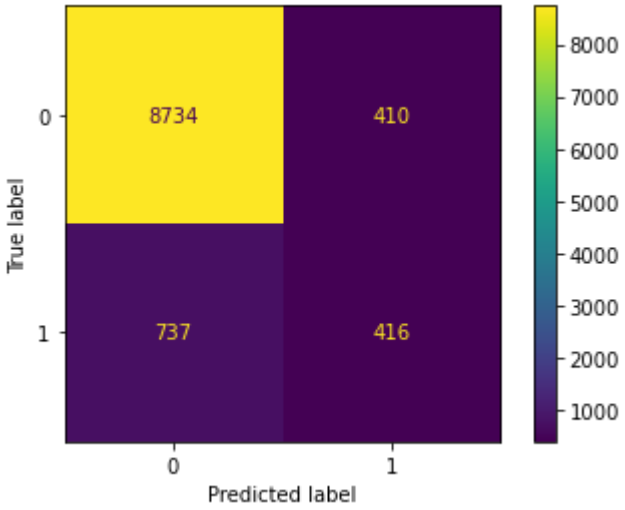
En el siguiente bloque se implementa un modelo de regresión logística donde se ha definido una función auxiliar propia en el cual se predice fijando el umbral internamente a 0.3 meiante la función: (model.predict_proba(x_test)[:,1] >=0.3) Con un eso se consigue por tanto detectar más casos positivos: para negocio puede ser importante porque me imagino que la idea es centrarse en el conjunto de la población con mayor probabilidad o tendencia a subscribirse a un depósito.

In [97]:

```
# Nuevas predcciones bajando el umbral a 0.3
# 1. Se importa nuestra librería de funciones auxiliares
import import_ipynb

%run functions.ipynb
# 2. Se importa el modelo de regresión logística
from sklearn.linear_model import LogisticRegression
# 3. Se importa librerías necesarias para entrenamiento, predicción e in
from sklearn import model_selection

from sklearn.metrics import plot_confusion_matrix,classification_report,
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
import matplotlib.pyplot as plt
import pandas as pd
# split train, test
from sklearn.model_selection import train_test_split
#from sklearn.metrics import precision_score

# 3. Lectura datos, preprocesado y preparación de datos
url='bank-additional-full.csv'
df_ref = pd.read_csv(url,sep=";")
rename=True
df2 =transform(df_ref,rename)
y = df2['deposit_n']
x = df2.drop('deposit_n', axis = 1)

# 4. split datos de entrenamiento y test
x_train,x_test,y_train,y_test = train_test_split(x,y, random_state=42)
# 5. Se invoca al modelo.
logreg_threshold = LogisticRegression(solver='liblinear', random_state =

# 6. Entrenamiento del modelo
name='Logistic Regression'
kfold = model_selection.KFold(n_splits=10, shuffle = True, random_state=
cv_results = model_selection.cross_val_score(logreg_threshold, x_train,
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)
logreg_threshold.fit(x_train, y_train)

# 7. Predicción con el nuevo threshold en 0.3 llamando a función propia
report_threshold(logreg_threshold)
```

```
Logistic Regression: 0.900521 (0.004396)
              precision    recall  f1-score   support
```

```
              precision    recall  f1-score   support

       False       0.96      0.92      0.94      9471
        True       0.36      0.50      0.42       826

    accuracy                           0.89     10297
   macro avg       0.66      0.71      0.68     10297
weighted avg       0.91      0.89      0.90     10297
```



Otra opción interesante podría ser obtener la probabilidad de que un determinado cliente se subscriba. Mediante el método predict_proba se obtiene el array de probabilidades. A modo de ejemplo se visualiza la probabilidad de los dos primeros datos visualizados y detallados en el último bloque. La matriz de probabilidades consiste en la probabilidad del no y la probabilidad del si para cada entrada de datos

In [98]:
```
# Visualización de probabilidades
# 1. Se obtiene las probabilidades del conjunto de test
predi=logreg_threshold.predict_proba(x_test)

# 2. Se imprime la matriz de los dos primeros clientes de test.
print(predi[:2])
#for i in range(len(Xnew)):
 #   print("X=%s, Predicted=%s" % (Xnew[i], ynew[i]))
```

```
[[0.89997877 0.10002123]
 [0.96940248 0.03059752]]
```

In [99]:
```
# Visualización de los dos primeros clientes de test de los que se ha pr
x_test.head(2)
```

Out[99]:

	age	n_contacts	days_from_last_campaign	n_past_contacts	ipc	ic
32884	1.628993	-0.565922	0.195414	1.671136	-1.179380	-1.23103
3169	1.437075	-0.204909	0.195414	-0.349494	0.722722	0.88644

2 rows × 38 columns

12. Ambitos de mejora

A continuación se muestran algunos ámbitos de mejora por donde se podría continuar con este proyecto:

1. Desarrollo de una plataforma o MLOPs donde se pueda automatizar el aprendizaje, realimentar nuevos datos y se mantenga la explotación.
2. Se podría investigar en un mejor preprocesado para aquellos modelos que son más sensibles a ello para ver si hay ámbito de mejora. Por ejemplo creacion de nuevas variables que puedan aportar valor a la predicción, mejor imputación o codificación de las variables ....
3. Implementación de una base de datos donde se actualicen nuevos datos y se actualicen los modelos in-situ
4. Para alcanzar mayor eficiencia en el modelado, se podría buscar una plataforma en la nube con procesadores potentes donde se pueda incluir mayores opciones de tuneado para observar los grados de mejora o google colab con la opcion GPU activada.
5. Estudiar la posibilidad de diferentes técnicas de desbalanceo en entornos desarrollo para observar si se obtienen buenas métricas.

   En función de los logros obtenidos, se podría proponer nuevas necesidades del proyecto de cara a futuro. Este proyecto está en una fase inicial, donde se ha realizado un estudio con un pequeño entorno de producción: existe un boraador en google colab: https://colab.research.google.com/drive/1m7xPfJjf9evEKV_1VTMeZVDlfH22OdII?hl=es#scrollTo=_arUhB1yE7v3 Se podría continuar con los àmbitos de mejora propuetos para seguir con su evolución.