In [8]:
```python
###importar librerías
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, accuracy_score, classification_report

from sklearn.decomposition import PCA, KernelPCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

from matplotlib.colors import ListedColormap
```

In [4]:
```python
###extraer base y visualizar
df = pd.read_csv('data1.csv')
df.head()
```

Out[4]:

	customerID	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	C
0	0002-ORFBO	9	1	0	DSL	0	1	0	1	1	0	O
1	0003-MKNFE	9	1	1	DSL	0	0	0	0	0	1	to
2	0004-TLHLJ	4	1	0	Fiber optic	0	0	1	0	0	0	to
3	0011-IGKFF	13	1	0	Fiber optic	0	1	1	0	1	1	to
4	0013-EXCHZ	3	1	0	Fiber optic	0	0	0	1	1	0	to

In [5]:
```python
###limpiar base
df = df.dropna()
df = df.drop_duplicates()

df = df[['tenure', 'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'Streaming

df.head()

df.describe()
```

Out[5]:

	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	PaperlessBilling	Mor
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	
mean	32.371149	0.903166	0.615505	0.720006	0.778220	0.777226	0.723555	0.817691	0.821241	0.592219	
std	24.559481	0.295752	0.656039	0.796885	0.778472	0.778826	0.795896	0.763212	0.761725	0.491457	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	9.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	29.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	
75%	55.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	
max	72.000000	1.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	1.000000	

In [10]:
```python
###definición de variables y separacion de train y test
df.info()

X = df[['tenure', 'PhoneService', 'MultipleLines','OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV
y =df['Churn']

X_train,X_test,y_train,y_test = train_test_split(X,y,random_state = 0, test_size=0.30)

df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7043 entries, 0 to 7042
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   tenure            7043 non-null   int64
 1   PhoneService      7043 non-null   int64
 2   MultipleLines     7043 non-null   int64
 3   OnlineSecurity    7043 non-null   int64
 4   OnlineBackup      7043 non-null   int64
 5   DeviceProtection  7043 non-null   int64
 6   TechSupport       7043 non-null   int64
 7   StreamingTV       7043 non-null   int64
 8   StreamingMovies   7043 non-null   int64
 9   PaperlessBilling  7043 non-null   int64
 10  MonthlyCharges    7043 non-null   float64
 11  Churn             7043 non-null   int64
dtypes: float64(1), int64(11)
memory usage: 715.3 KB
```

Out[10]:

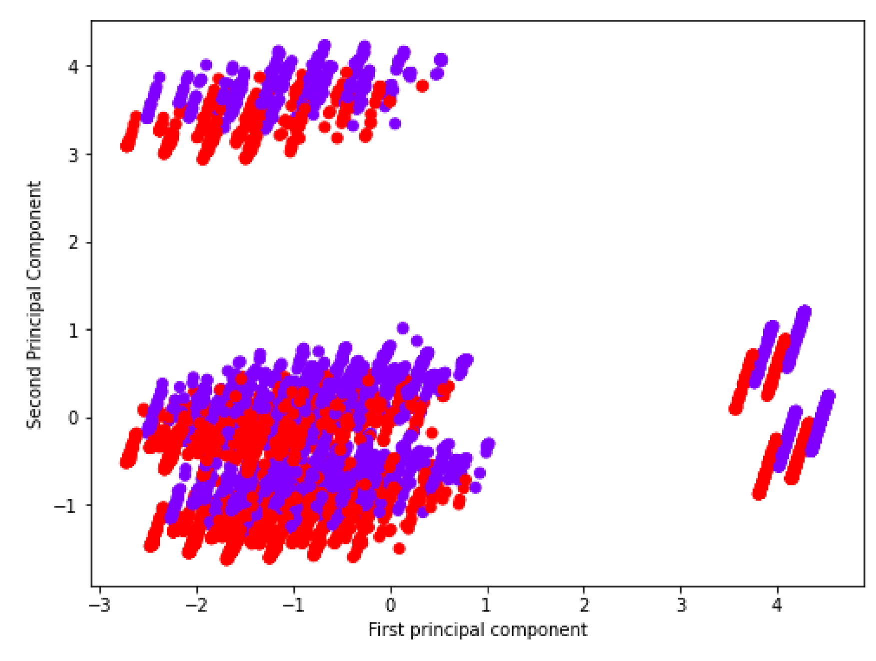
	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	PaperlessBilling	MonthlyCharg
0	9	1	0	0	1	0	1	1	0	1	65
1	9	1	1	0	0	0	0	0	1	0	59
2	4	1	0	0	0	1	0	0	0	1	73
3	13	1	0	0	1	1	0	1	1	1	98
4	3	1	0	0	0	0	1	1	0	1	83

In [12]:
```python
###transformar datos para la unificación de variables normalización
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

scaler = StandardScaler()

scaler.fit(df)
scaled_data = scaler.transform(df)
```

In [13]:
```python
###reducir el número de características
pca = PCA(n_components=2)

pca.fit(scaled_data)
X_pca = pca.transform(scaled_data)

###ver dimensiones shape

print(scaled_data.shape)

print('Reducción de dimensionalidad con shape')
print(X_pca.shape)




#pca.fit(X)
```

```
(7043, 12)
Reducción de dimensionalidad con shape
(7043, 2)
```

In [14]:
```python
###Gráfica
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0],X_pca[:,1],c=y,cmap='rainbow')
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
plt.show()
```



In [16]:
```python
###reducir variable x
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

In [17]:
```python
# X_pca = pca.transform(X)
# print(X.shape)
# print(X_pca.shape)
X.head()
```

Out[17]:

	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	PaperlessBilling	MonthlyCharg
0	9	1	0	0	1	0	1	1	0	1	65
1	9	1	1	0	0	0	0	0	1	0	59
2	4	1	0	0	0	1	0	0	0	1	73
3	13	1	0	0	1	1	0	1	1	1	98
4	3	1	0	0	0	0	1	1	0	1	83

In [18]:
```python
#visualizar nuestras reducciones
print('Componentes Principales', pca.components_)
print('Varianza explicada', pca.explained_variance_)
print('Tasa de varianza explicada', pca.explained_variance_ratio_)
```

```
Componentes Principales [[1. 0.]
 [0. 1.]]
Varianza explicada [5.35162717 1.70076507]
Tasa de varianza explicada [0.75883856 0.24116144]
```

In [22]:
```python
#Realización del modelo
clf = LogisticRegression(random_state=0)
model = clf.fit(X_train, y_train)

print(model.intercept_)
print(model.coef_)
print('Matriz de probabilidades')
print('Prob = 0, Prob = 1')
print(model.predict_proba(X_train))

print('-------------------------')
print(model.score(X_train, y_train))
```

```
[0.00802586]
[[-0.18263634 -0.1463361 ]]
Matriz de probabilidades
Prob = 0, Prob = 1
[[0.4996591  0.5003409 ]
 [0.69997692 0.30002308]
 [0.60012895 0.39987105]
 ...
 [0.41933397 0.58066603]
 [0.67152887 0.32847113]
 [0.49707887 0.50292113]]
-------------------------
0.585395537525355
```

In [23]:
```python
print(X_train)
print('--------- y_train')
print(y_train)
```

```
[[-0.08316177  0.14931812]
 [ 4.11608836  0.7070469 ]
 [-0.83915568  3.87661437]
 ...
 [-1.05570872 -0.85195292]
 [ 4.29876393 -0.42352216]
 [ 0.45332162 -0.59077563]]
--------- y_train
3580    0
2364    0
6813    0
789     1
561     0
       ..
4931    1
3264    1
1653    0
2607    1
2732    0
Name: Churn, Length: 4930, dtype: int64
```

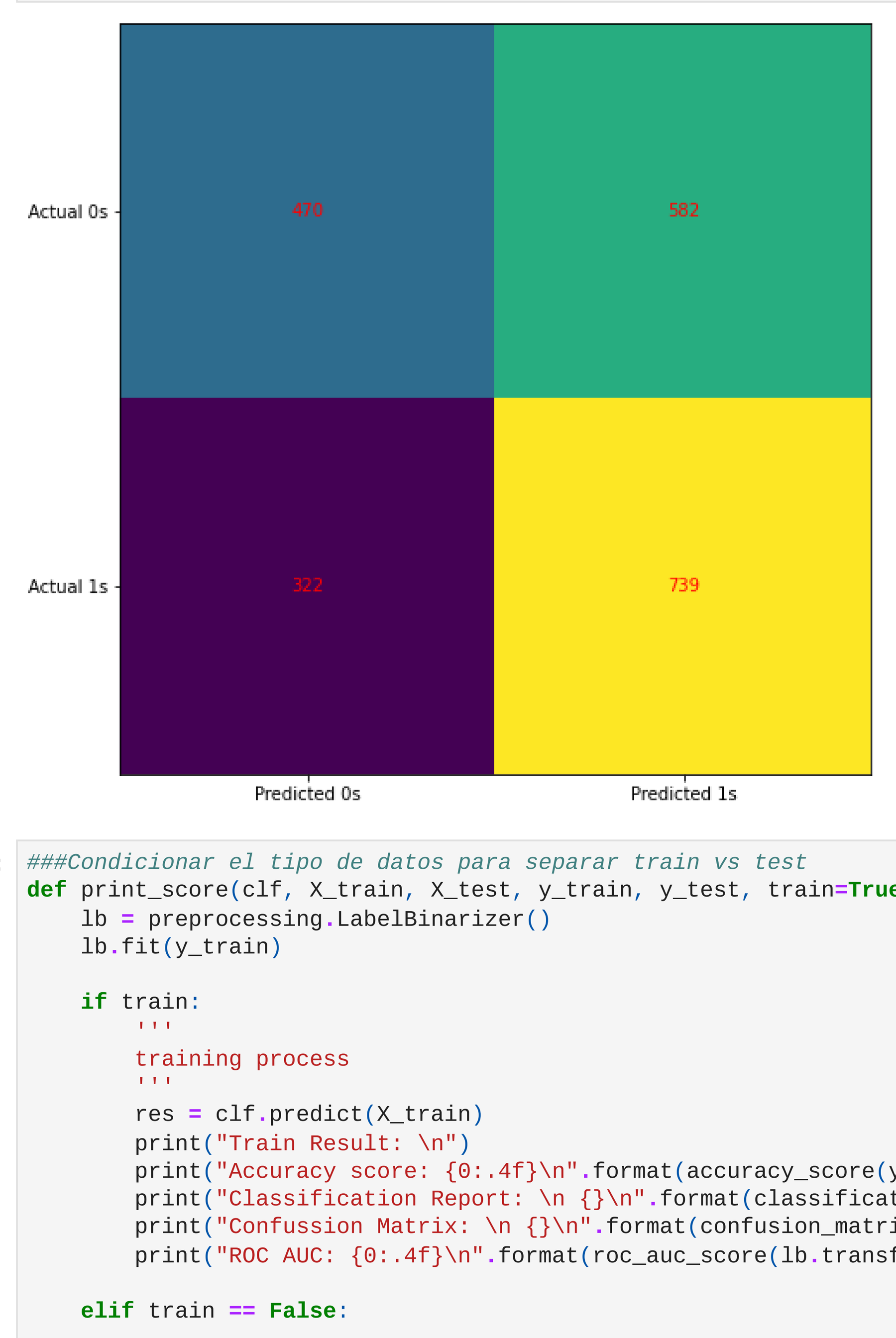
In [24]:
```python
###Predicción
y_pred = clf.predict(X_test)
print(y_pred)
```

```
[1 1 0 ... 1 0 0]
```

In [25]:
```python
###Matriz de confusión
cm = confusion_matrix(y_test, y_pred)
cm
print(cm)
```

```
[[470 582]
 [322 739]]
```

In [26]:
```python
###Graficar matriz de confusión
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
plt.show()
```



In [28]:
```python
###Condicionar el tipo de datos para separar train vs test
def print_score(clf, X_train, X_test, y_train, y_test, train=True):
    lb = preprocessing.LabelBinarizer()
    lb.fit(y_train)

    if train:
        '''
        training process
        '''
        res = clf.predict(X_train)
        print("Train Result: \n")
        print("Accuracy score: {0:.4f}\n".format(accuracy_score(y_train, res)))
        print("Classification Report: \n {}\n".format(classification_report(y_train, res)))
        print("Confussion Matrix: \n {}\n".format(confusion_matrix(y_train, res)))
        print("ROC AUC: {0:.4f}\n".format(roc_auc_score(lb.transform(y_train), lb.transform(res))))

    elif train == False:

        res_test = clf.predict(X_test)
        print("Train Result: \n")
        print("Accuracy score: {0:.4f}\n".format(accuracy_score(y_test, res_test)))
        print("Classification Report: \n {}\n".format(classification_report(y_test, res_test)))
        print("Confussion Matrix: \n {}\n".format(confusion_matrix(y_test, res_test)))
        print("ROC AUC: {0:.4f}\n".format(roc_auc_score(lb.transform(y_test), lb.transform(res_test))))
```

In [29]:
```python
###Imprimir los resultados
print_score(clf, X_train, X_test, y_train, y_test, train=True )
```

```
Train Result:

Accuracy score: 0.5854

Classification Report:
               precision    recall  f1-score   support

           0       0.61      0.45      0.52      2449
           1       0.57      0.72      0.63      2481

    accuracy                           0.59      4930
   macro avg       0.59      0.58      0.58      4930
weighted avg       0.59      0.59      0.58      4930


Confussion Matrix:
 [[1109 1340]
 [ 704 1777]]

ROC AUC: 0.5845
```

In [ ]: