{

"cells": [

{

"cell\_type": "markdown",

"metadata": {

"id": "uOxsWL4fyQ2J"

},

"source": [

"# Aula 1 - Clasificación: ¿Cómo funciona?"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "vuZVqaVCyQ2M"

},

"source": [

"## 1.1 - Importando los datos"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "EcFx0fP0yQ2N"

},

"outputs": [],

"source": [

"import pandas as pd"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "jMxgxTXnyQ2O"

},

"outputs": [],

"source": [

"datos = pd.read\_csv('Customer.csv')"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "neE16a7wyQ2O"

},

"outputs": [],

"source": [

"datos.shape"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "k\_QTfXOSyQ2P"

},

"outputs": [],

"source": [

"datos.head()"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "V8lNCeRnyQ2P"

},

"source": [

"## 1.2 - Analizando las Variables"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "uQLvJoZ0yQ2Q"

},

"outputs": [],

"source": [

"#Modificación de forma manual\n",

"diccionario = {'Si': 1,\n",

" 'No': 0}\n",

"\n",

"datosmodificados = datos[['Conyuge', 'Dependientes', 'TelefonoFijo', 'PagoOnline', 'Churn']].replace(diccionario)\n",

"datosmodificados.head()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "0Dc3-8wayQ2R"

},

"outputs": [],

"source": [

"#Transformación con get\_dummies\n",

"dummie\_datos = pd.get\_dummies(datos.drop(['Conyuge', 'Dependientes', 'TelefonoFijo', 'PagoOnline', 'Churn'],\n",

" axis=1))\n",

"\n",

"#Unión de los datos transformados con los que ya teníamos\n",

"datos\_final = pd.concat([datosmodificados, dummie\_datos], axis=1)"

]

},

{

"cell\_type": "code",

"source": [

"datos\_final.head()"

],

"metadata": {

"id": "UmQrjUnh08vu"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## 1.3 - Definición Formal"

],

"metadata": {

"id": "D1YLHA3h37s1"

}

},

{

"cell\_type": "markdown",

"source": [

"Informaciones para la clasificación:\n",

"\n",

"$X$ = inputs (datos de entrada)\n",

"\n",

"$y$ = outputs (datos de salida)"

],

"metadata": {

"id": "sSgnt7QWM5rm"

}

},

{

"cell\_type": "code",

"source": [

"#TIP\n",

"pd.set\_option('display.max\_columns', 39)"

],

"metadata": {

"id": "k\_o0S4UyOqzd"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"datos\_final.head()"

],

"metadata": {

"id": "oiHVlSC737NE"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"\n",

"$y\_i$ = $f(x\_i)$"

],

"metadata": {

"id": "mlxxn4VYTQnV"

}

},

{

"cell\_type": "code",

"source": [

"Xmaria = [[0,0,1,1,0,0,39.90,1,0,0,0,1,0,1,0,0,0,0,1,1,1,0,0,1,0,1,0,0,0,0,1,0,0,1,0,0,0,1]]"

],

"metadata": {

"id": "JVqSH1yp4Cq2"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#ymaria = ?"

],

"metadata": {

"id": "1Btl3Wct4CoN"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"Nuevo par de informaciones = ($Xmaria$, $ymaria$)"

],

"metadata": {

"id": "n4nyxwPFTncE"

}

},

{

"cell\_type": "markdown",

"source": [

"## 1.4 - Balanceamiento de los datos"

],

"metadata": {

"id": "zCcuXt0FZfNe"

}

},

{

"cell\_type": "code",

"source": [

"#variable target está desbalanceada\n",

"import seaborn as sns\n",

"%matplotlib inline\n",

"ax = sns.countplot(x='Churn', data=datos\_final)"

],

"metadata": {

"id": "PZ98-642QERQ"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"datos\_final.Churn.value\_counts()"

],

"metadata": {

"id": "tDmNwxUGQt0U"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#biblioteca para balancear los datos utilizando over\_sampling\n",

"from imblearn.over\_sampling import SMOTE"

],

"metadata": {

"id": "TcFY4SXuZyIO"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#dividiendo los datos en características y target\n",

"X = datos\_final.drop('Churn', axis = 1)\n",

"y = datos\_final['Churn']"

],

"metadata": {

"id": "0-I2aDVhQUdu"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"smt = SMOTE(random\_state=123)\n",

"X, y = smt.fit\_resample(X, y)"

],

"metadata": {

"id": "5lIY03boZyBN"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#unión de los datos balanceados\n",

"datos\_final = pd.concat([X, y], axis=1)"

],

"metadata": {

"id": "bnlqzLVUZl0u"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#verificación 1 - unión de los datos\n",

"datos\_final.head(2)"

],

"metadata": {

"id": "GQj3bEEOZ42e"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#verificación 2 - balanceamiento\n",

"ax = sns.countplot(x='Churn', data=datos\_final)"

],

"metadata": {

"id": "jSzaHOlyZ7r-"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"datos\_final.Churn.value\_counts()"

],

"metadata": {

"id": "Zx0HYHd3RGl7"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"# Aula 2 - Método baseado en la proximidad"

],

"metadata": {

"id": "\_d4jEXubcPuO"

}

},

{

"cell\_type": "markdown",

"source": [

"## 2.1 - Modelo K-nearest neighbors (KNN)\n",

"\n",

"(PPT)"

],

"metadata": {

"id": "PVGb9U19cW4V"

}

},

{

"cell\_type": "markdown",

"source": [

"## 2.2 - KNN en la práctica"

],

"metadata": {

"id": "MvIfmyE2ckVV"

}

},

{

"cell\_type": "code",

"source": [

"Xmaria"

],

"metadata": {

"id": "blRWRPdXcWJ1"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#ymaria = ?"

],

"metadata": {

"id": "tdwhZOphjbIg"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#División en inputs y outputs\n",

"X = datos\_final.drop('Churn', axis = 1)\n",

"y = datos\_final['Churn']"

],

"metadata": {

"id": "h0re3ABxjlLW"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#biblioteca para padronizar los datos\n",

"from sklearn.preprocessing import StandardScaler"

],

"metadata": {

"id": "cw\_-29nlkLMV"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"norm = StandardScaler()\n",

"X\_normalizado = norm.fit\_transform(X)\n",

"X\_normalizado"

],

"metadata": {

"id": "zGYhtla\_kgpd"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"X\_normalizado[0]"

],

"metadata": {

"id": "Cs-ivgv7qsKL"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"Xmaria\_normalizado = norm.transform(pd.DataFrame(Xmaria, columns = X.columns))\n",

"Xmaria\_normalizado"

],

"metadata": {

"id": "Ll6FnFO-kosd"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"Distancia Euclidiana:\n",

"\n",

"$\\sqrt{\\sum\_{i=1}^k(a\_{i}-b\_{i})^2}$\n"

],

"metadata": {

"id": "pgU6OxezlB\_F"

}

},

{

"cell\_type": "code",

"source": [

"import numpy as np"

],

"metadata": {

"id": "SbxwDdiIk36F"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"a = Xmaria\_normalizado"

],

"metadata": {

"id": "9oywjidemxXl"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"b = X\_normalizado[0]"

],

"metadata": {

"id": "JL85j369li4d"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#1 - comenzamos restando\n",

"a - b"

],

"metadata": {

"id": "mmhEqixanDgF"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#2 - después realizamos la exponenciación\n",

"np.square(a-b)"

],

"metadata": {

"id": "\_aBlq3wPnF49"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#3 - realizamos la suma\n",

"np.sum(np.square(a-b))"

],

"metadata": {

"id": "\_bDEeU\_znIxH"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#4 - finalmente obtenemos la raiz cuadrada y tenemos nuestra distancia\n",

"np.sqrt(103.36325779671671)"

],

"metadata": {

"id": "o9NJQSP-nMUm"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## 2.3 - Implementando el modelo"

],

"metadata": {

"id": "\_h0pFQfxwYW-"

}

},

{

"cell\_type": "code",

"source": [

"#biblioteca para división de los datos\n",

"from sklearn.model\_selection import train\_test\_split"

],

"metadata": {

"id": "XtvZ-tMiwjN2"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_normalizado, y, test\_size=0.3, random\_state=123)"

],

"metadata": {

"id": "lWaKqJrTwxE-"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"### Entrenamiento y prueba"

],

"metadata": {

"id": "LOKzwv4Uw0u7"

}

},

{

"cell\_type": "code",

"source": [

"#biblioteca para crear el modelo de machine learning\n",

"from sklearn.neighbors import KNeighborsClassifier"

],

"metadata": {

"id": "g6-t-BpEwy3G"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#iniciar el modelo (creamos el modelo) - por default son 5 vecinos\n",

"knn = KNeighborsClassifier(metric='euclidean')"

],

"metadata": {

"id": "qFs4VHnhw4-O"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#entrenando el modelo con los datos de entrenamiento\n",

"knn.fit(X\_train, y\_train)"

],

"metadata": {

"id": "H05zbtgIw6gt"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#probando el modelo con los datos de prueba\n",

"prediccion\_knn = knn.predict(X\_test)"

],

"metadata": {

"id": "2LrZiJbvw-Mf"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"prediccion\_knn"

],

"metadata": {

"id": "dirCpdIO1MZ9"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"# Aula 3 - Método probabilístico"

],

"metadata": {

"id": "EKHFh1Pr7G40"

}

},

{

"cell\_type": "markdown",

"source": [

"## 3.1 - Teorema de Naive Bayes\n",

"\n",

"(PPT)"

],

"metadata": {

"id": "b\_EAQ1nf7VUu"

}

},

{

"cell\_type": "markdown",

"source": [

"## 3.2 - Modelo Bernoulli Naive Bayes\n",

"\n",

"(PPT)"

],

"metadata": {

"id": "02cUbMWf7dEN"

}

},

{

"cell\_type": "markdown",

"source": [

"## 3.3 - Entrenamiento y prueba"

],

"metadata": {

"id": "zIhV625O7lf9"

}

},

{

"cell\_type": "code",

"source": [

"#biblioteca para crear el modelo de machine learning\n",

"from sklearn.naive\_bayes import BernoulliNB"

],

"metadata": {

"id": "vV4i1JC37xJN"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#escojo utilizar mediana, porque es el valor central de nuestros datos ordenados\n",

"mediana = np.median(X\_train)\n",

"mediana"

],

"metadata": {

"id": "AEzm1GSZ7tZZ"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#Binarizando los recursos usando la mediana\n",

"X\_train\_binarizado = np.where(X\_train > mediana, 1, 0)"

],

"metadata": {

"id": "X0HEgH1UXG1l"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"X\_train\_binarizado"

],

"metadata": {

"id": "e5HdbyA27cUJ"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"y\_train"

],

"metadata": {

"id": "HevdnXNV7rPg"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#creamos el modelo\n",

"bnb = BernoulliNB()"

],

"metadata": {

"id": "KnTnDve07ylb"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#entrenando el modelo\n",

"bnb.fit(X\_train\_binarizado, y\_train)"

],

"metadata": {

"id": "qIIT0uX273SN"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#Binarizando la base de prueba\n",

"X\_test\_binarizado = np.where(X\_test > np.median(X\_test), 1, 0)"

],

"metadata": {

"id": "2v7NF1BipX6w"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#probando el modelo\n",

"prediccion\_BNb = bnb.predict(X\_test\_binarizado)"

],

"metadata": {

"id": "BpCOWhx674G6"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"prediccion\_BNb"

],

"metadata": {

"id": "KIS6Syvf79ou"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"# Aula 4 - Método Simbólico"

],

"metadata": {

"id": "I0QZ6\_rY00NG"

}

},

{

"cell\_type": "markdown",

"source": [

"## 4.1 - ¿Qué es un árbol de decisión?\n",

"\n",

"(PPT)"

],

"metadata": {

"id": "VglEIUuC2jkO"

}

},

{

"cell\_type": "markdown",

"source": [

"## 4.2 - ¿Cómo funciona un árbol de decisión?\n",

"\n",

"(PPT)"

],

"metadata": {

"id": "jdtLgDk3sAQZ"

}

},

{

"cell\_type": "markdown",

"source": [

"## 4.3 - Implementando el modelo"

],

"metadata": {

"id": "cYFuWWnTsJLP"

}

},

{

"cell\_type": "code",

"source": [

"#biblioteca para crear el modelo de machine learning\n",

"from sklearn.tree import DecisionTreeClassifier"

],

"metadata": {

"id": "VZBVqKxE2i10"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#iniciando el modelo\n",

"dtc = DecisionTreeClassifier(criterion='entropy', random\_state=42)"

],

"metadata": {

"id": "AR0fPU6qsPmg"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#entrenando el modelo\n",

"dtc.fit(X\_train, y\_train)"

],

"metadata": {

"id": "zG5vDatzsV7D"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#verificando la importancia de cada atributo\n",

"dtc.feature\_importances\_"

],

"metadata": {

"id": "Gx2S564TiNqI"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"prediccion\_ArbolDecision = dtc.predict(X\_test)"

],

"metadata": {

"id": "1N3ogMwXsYLJ"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"prediccion\_ArbolDecision"

],

"metadata": {

"id": "8yWcv1rCsbIT"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"# Aula 5 - Validación de los modelos"

],

"metadata": {

"id": "lfmj\_l6ztF5J"

}

},

{

"cell\_type": "markdown",

"source": [

"## 5.1 - Matriz de confusión\n",

"\n"

],

"metadata": {

"id": "5LcpGgohtOM5"

}

},

{

"cell\_type": "markdown",

"source": [

"![image.png](https://raw.githubusercontent.com/ElProfeAlejo/machine\_learning\_clasificacion/main/imagen\_5\_1.png)"

],

"metadata": {

"id": "BbHrkrAnxhmv"

}

},

{

"cell\_type": "code",

"source": [

"from sklearn.metrics import confusion\_matrix"

],

"metadata": {

"id": "dIhkghN1tM3d"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"print(confusion\_matrix(y\_test, prediccion\_knn))"

],

"metadata": {

"id": "vy5vfPQ7mnmP"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"print(confusion\_matrix(y\_test, prediccion\_BNb))"

],

"metadata": {

"id": "Y\_TkgT9pmnj3"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"print(confusion\_matrix(y\_test, prediccion\_ArbolDecision))"

],

"metadata": {

"id": "hNAbk\_Pimnhw"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## 5.2 - Accuracy\n",

"A partir del cálculo de la matriz de confusión, podemos inferir otras métricas, como el accuracy.\n",

"\n",

"\n",

"\n",

"\n",

"$ACC$ = ${TP + TN \\over TP + FP + TN + FN}$"

],

"metadata": {

"id": "sQb7vnC12XUf"

}

},

{

"cell\_type": "code",

"source": [

"from sklearn.metrics import accuracy\_score"

],

"metadata": {

"id": "Ng67mpWk2af1"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo KNN\n",

"print(accuracy\_score(y\_test, prediccion\_knn))"

],

"metadata": {

"id": "gp4Purjf2kZe"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo Bernoulli de Naive Bayes\n",

"print(accuracy\_score(y\_test, prediccion\_BNb))"

],

"metadata": {

"id": "7JrJXLJr2loH"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo Arbol de Decisión\n",

"print(accuracy\_score(y\_test, prediccion\_ArbolDecision))"

],

"metadata": {

"id": "BkvmkoN92nBX"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## 5.3 - Precisión\n",

"\n",

"Otra métrica importante es la precisión, que calcula cuántos se clasificaron correctamente como positivos ($TP$).\n",

"\n",

"$PS$ = ${TP \\over TP + FP}$"

],

"metadata": {

"id": "kfzcKhw42tmX"

}

},

{

"cell\_type": "code",

"source": [

"from sklearn.metrics import precision\_score"

],

"metadata": {

"id": "BM4AbIm62wP6"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo KNN\n",

"print(precision\_score(y\_test, prediccion\_knn))"

],

"metadata": {

"id": "hBD4K9r523g4"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo Bernoulli de Naive Bayes\n",

"print(precision\_score(y\_test, prediccion\_BNb))"

],

"metadata": {

"id": "mUJumEYp24rf"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo Arbol de Decisión\n",

"print(precision\_score(y\_test, prediccion\_ArbolDecision))"

],

"metadata": {

"id": "ST2dcOPJ25IS"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## 5.4 - Recall\n",

"\n",

"Otra métrica es el Recall o sensibilidad, calcula qué tan bueno es el modelo para clasificar correctamente un resultado positivo ($TP$).\n",

"\n",

"$RC$ = ${TP \\over TP + FN}$"

],

"metadata": {

"id": "TW59AoyU3DvW"

}

},

{

"cell\_type": "code",

"source": [

"from sklearn.metrics import recall\_score"

],

"metadata": {

"id": "SGX6E4Pi3LCt"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo KNN\n",

"print(recall\_score(y\_test, prediccion\_knn))"

],

"metadata": {

"id": "CVEe2W\_B3OTc"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo Bernoulli de Naive Bayes\n",

"print(recall\_score(y\_test, prediccion\_BNb))"

],

"metadata": {

"id": "edPMCLA93Pw\_"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#modelo Arbol de Decisión\n",

"print(recall\_score(y\_test, prediccion\_ArbolDecision))"

],

"metadata": {

"id": "4Wk5-4ej3Q\_3"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## 5.5 - Escogiendo el mejor modelo\n"

],

"metadata": {

"id": "4O7N26Fg3XwI"

}

},

{

"cell\_type": "code",

"source": [

"#Ejemplo - análisis de las precisiones previamente calculadas\n",

"print('Modelo KNN: ', precision\_score(y\_test, prediccion\_knn))\n",

"print('Modelo Bernoulli de Naive Bayes: ', precision\_score(y\_test, prediccion\_BNb))\n",

"print('Modelo Arbol de Decisión: ', precision\_score(y\_test, prediccion\_ArbolDecision))"

],

"metadata": {

"id": "RJ5Ls9lx3pfI"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"#Probando el mejor modelo para Maria\n",

"prediccion\_maria = knn.predict(Xmaria\_normalizado)\n",

"diccionario = {'Si': 1, 'No': 0}\n",

"\n",

"clave\_encontrada = next((clave for clave, valor in diccionario.items() if valor == prediccion\_maria[0]), None)\n",

"print(f\"La probabilidad de que Maria se convierta en Churn es: {clave\_encontrada}\")"

],

"metadata": {

"id": "BmhZXFOFganM"

},

"execution\_count": null,

"outputs": []

}

],

"metadata": {

"interpreter": {

"hash": "df87b94f50c6b8081a9a425f6ade537f06441a824b7d68371eb29c5a5eb2838b"

},

"kernelspec": {

"display\_name": "Python 3.10.4 64-bit",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.10.4"

},

"orig\_nbformat": 4,

"colab": {

"provenance": [],

"collapsed\_sections": [

"uOxsWL4fyQ2J",

"\_d4jEXubcPuO",

"EKHFh1Pr7G40",

"I0QZ6\_rY00NG",

"lfmj\_l6ztF5J"

]

}

},

"nbformat": 4,

"nbformat\_minor": 0

}