Universidad Industrial de Santander

75 Aniversario UIS 1948 - 2023

Legado académico y cultural de los santandereanos

Pronóstico de Aceptación de Depósito a Plazo

Freddy Santiago Galán Figueroa

Proyecto Final IA 2023-2

#LaUISqueQueremos

UC Irvine Machine Learning Repository

#LaUISqueQueremos

Universidad Industrial de Santander

Bank Marketing

Donated on 2/13/2012

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate	Business	Classification
Feature Type	**# Instances**	**# Features**
Categorical, Integer	45211	16

Additional Information

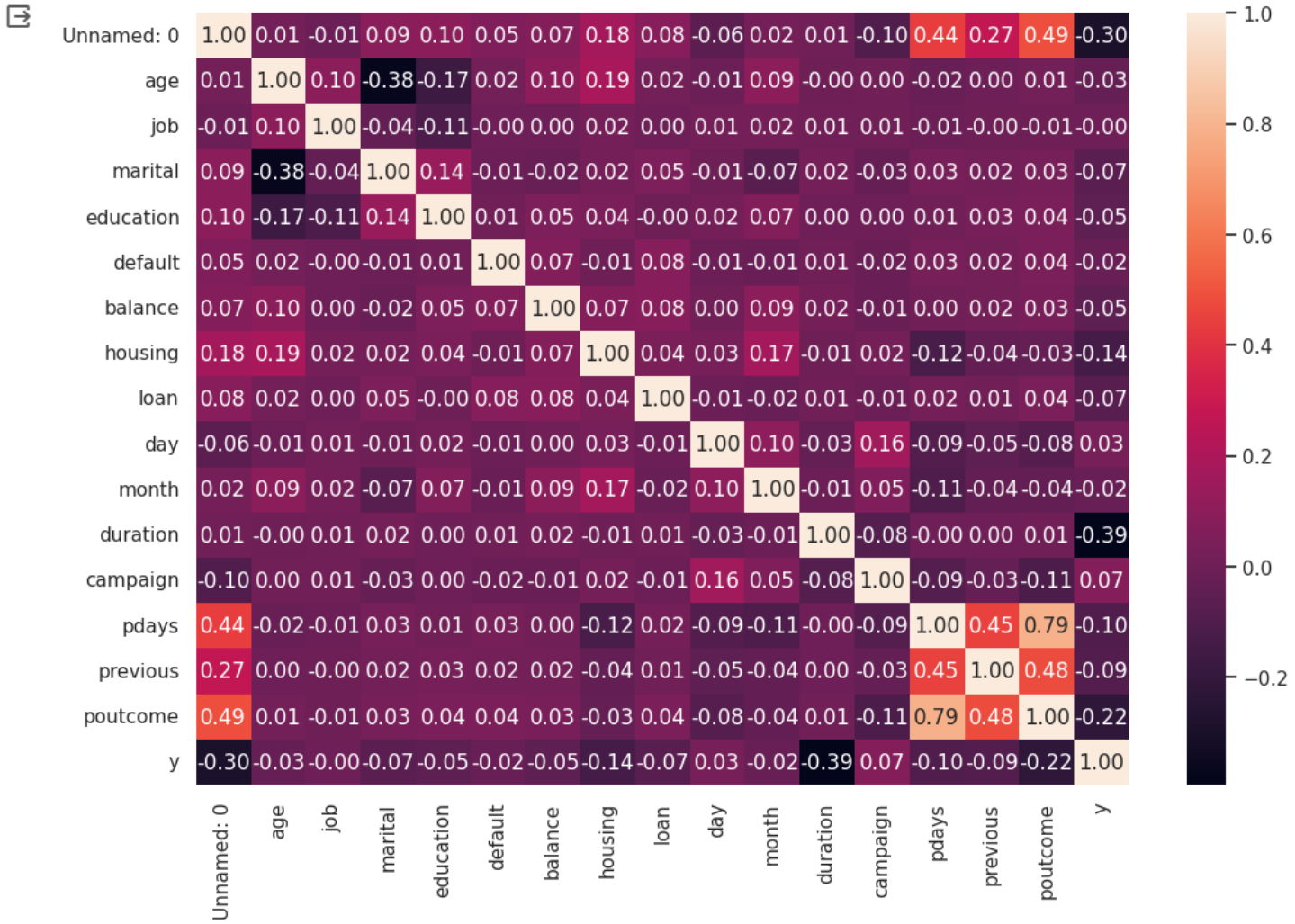
The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

EDA

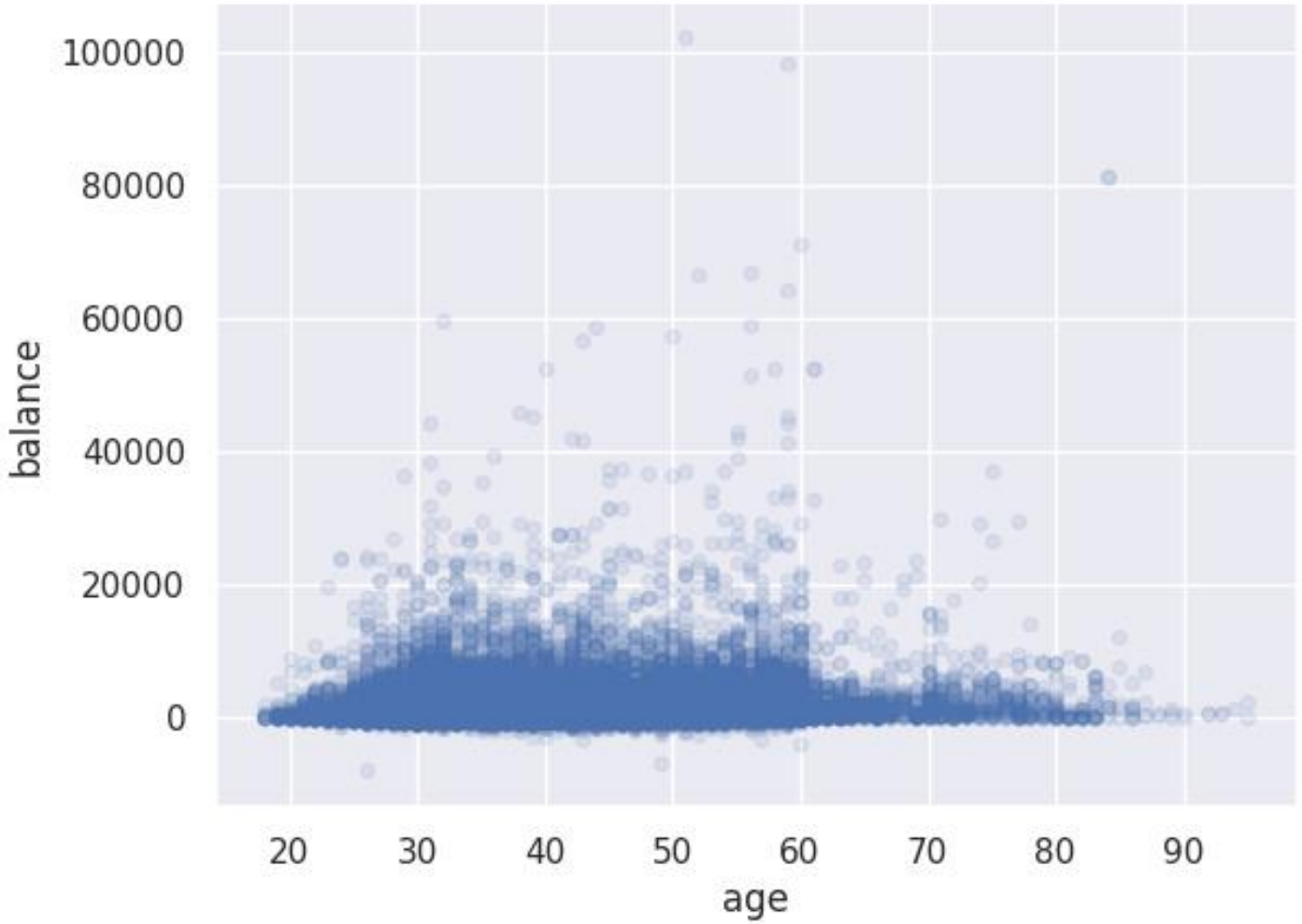

```
[5]  1 a =  pd.read_csv("datamod.csv", sep=",")
     2 a.head(10)
```

	Unnamed: 0	age	job	marital	education	default	balance	housing	loan	day	month	duration	campaign	pdays	previous	poutcome	y
0	0	58	4	1	4	2	2143	1	2	5	5	261	1	-1	0	1	2
1	1	44	11	3	3	2	29	1	2	5	5	151	1	-1	0	1	2
2	2	33	6	1	3	2	2	1	1	5	5	76	1	-1	0	1	2
3	3	47	8	1	1	2	1506	1	2	5	5	92	1	-1	0	1	2
4	4	33	1	3	1	2	1	2	2	5	5	198	1	-1	0	1	2
5	5	35	4	1	4	2	231	1	2	5	5	139	1	-1	0	1	2
6	6	28	4	3	4	2	447	1	1	5	5	217	1	-1	0	1	2
7	7	42	6	2	4	1	2	1	2	5	5	380	1	-1	0	1	2
8	8	58	10	1	2	2	121	1	2	5	5	50	1	-1	0	1	2
9	9	43	11	3	3	2	593	1	2	5	5	55	1	-1	0	1	2

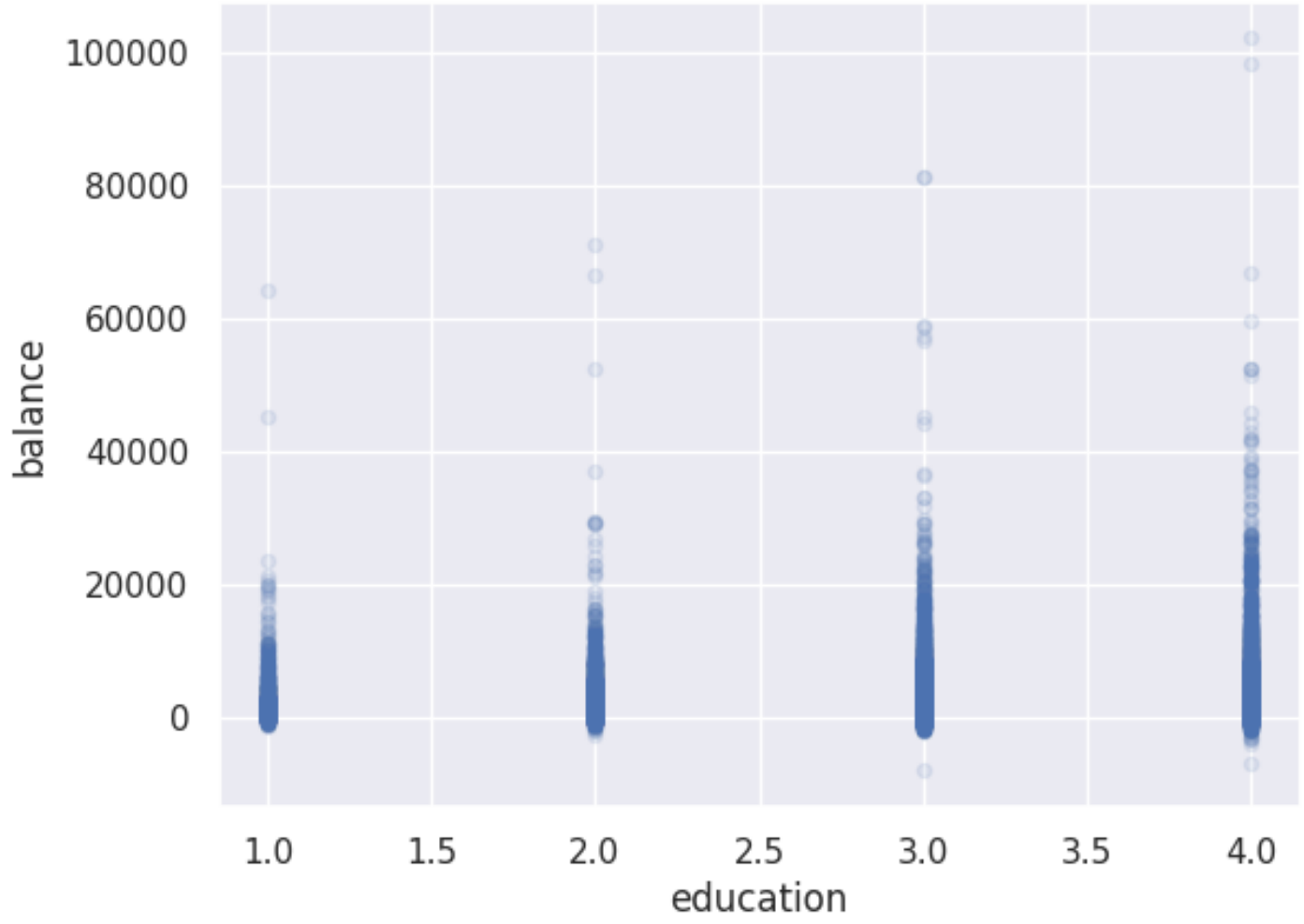
EDA



EDA



EDA



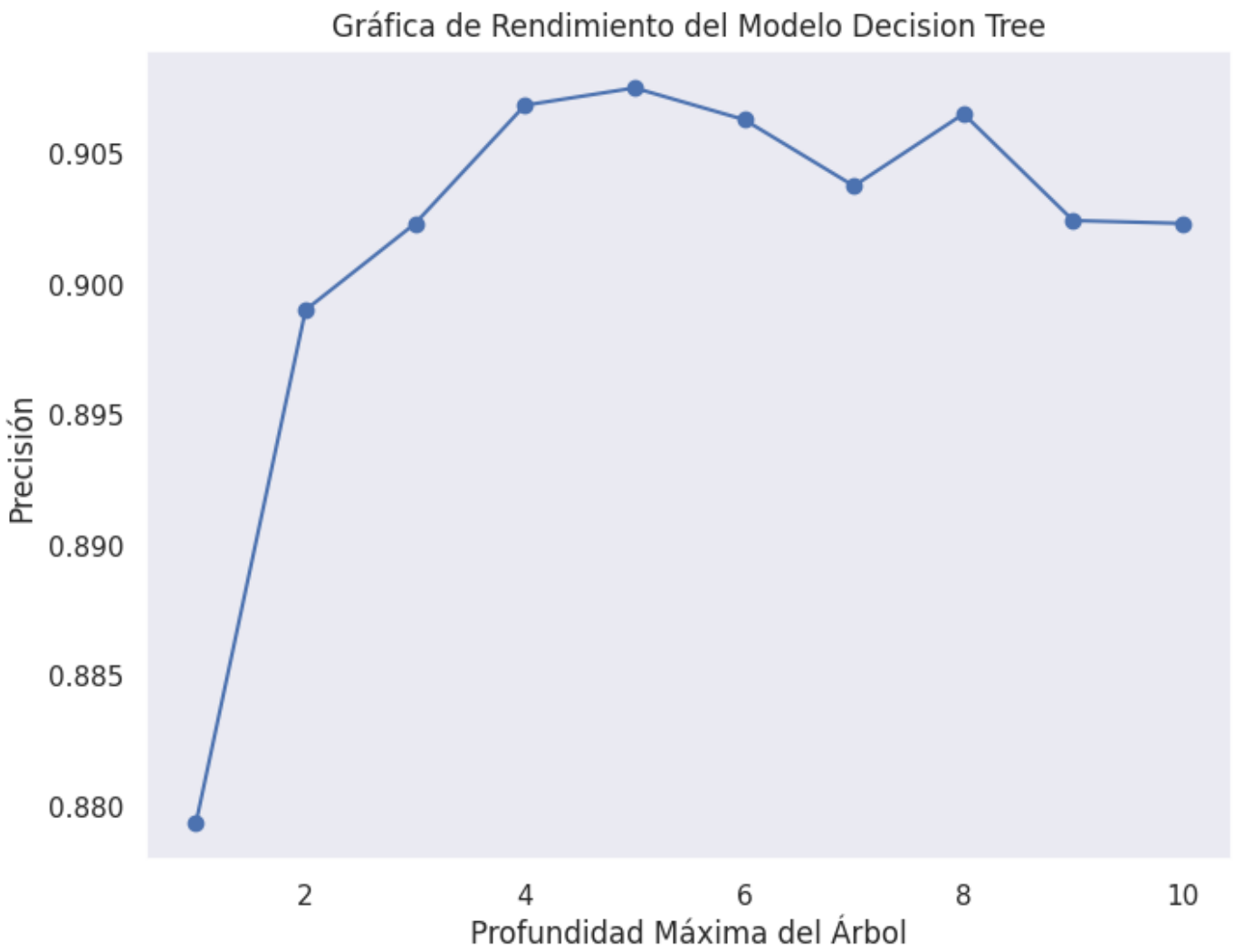
Modelos de clasificación


```python
#@title **DECISION TREE - CLASIFICACION**
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

X = a.drop(columns=['y'])
y = a['y']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=21)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
max_depth_values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
accuracies = []

for max_depth in max_depth_values:

    est = DecisionTreeClassifier(max_depth=max_depth)
    est.fit(X_train, y_train)
    y_pred = est.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
```

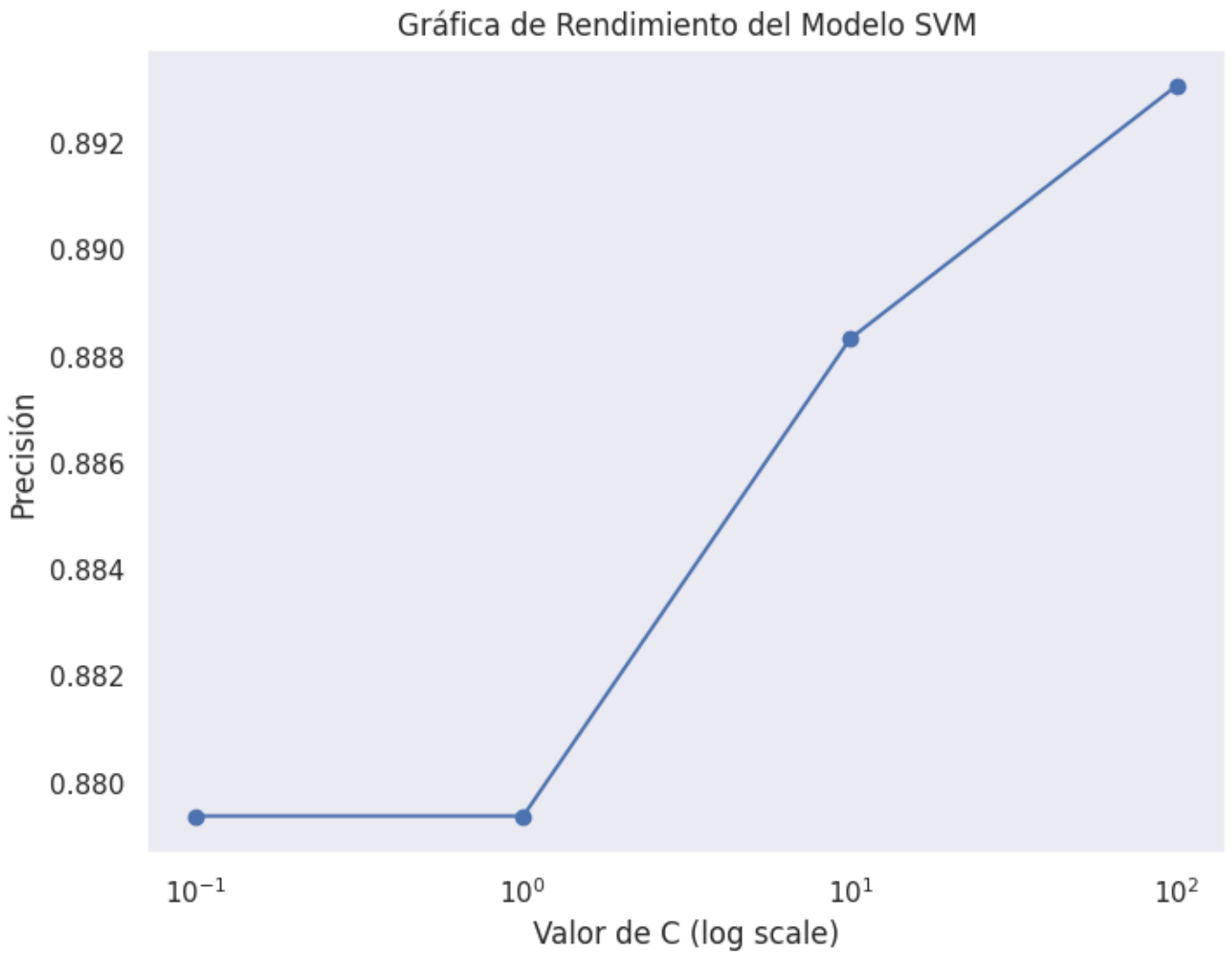
Modelos de clasificación


```python
#@title **SVM - CLASIFICACION**
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

C_values = [0.1, 1, 10, 100]

accuracies = []

for C in C_values:

    est = SVC(C=C)

    est.fit(X_train, y_train)

    y_pred = est.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
```

Modelos de clasificación



Modelos de clasificación

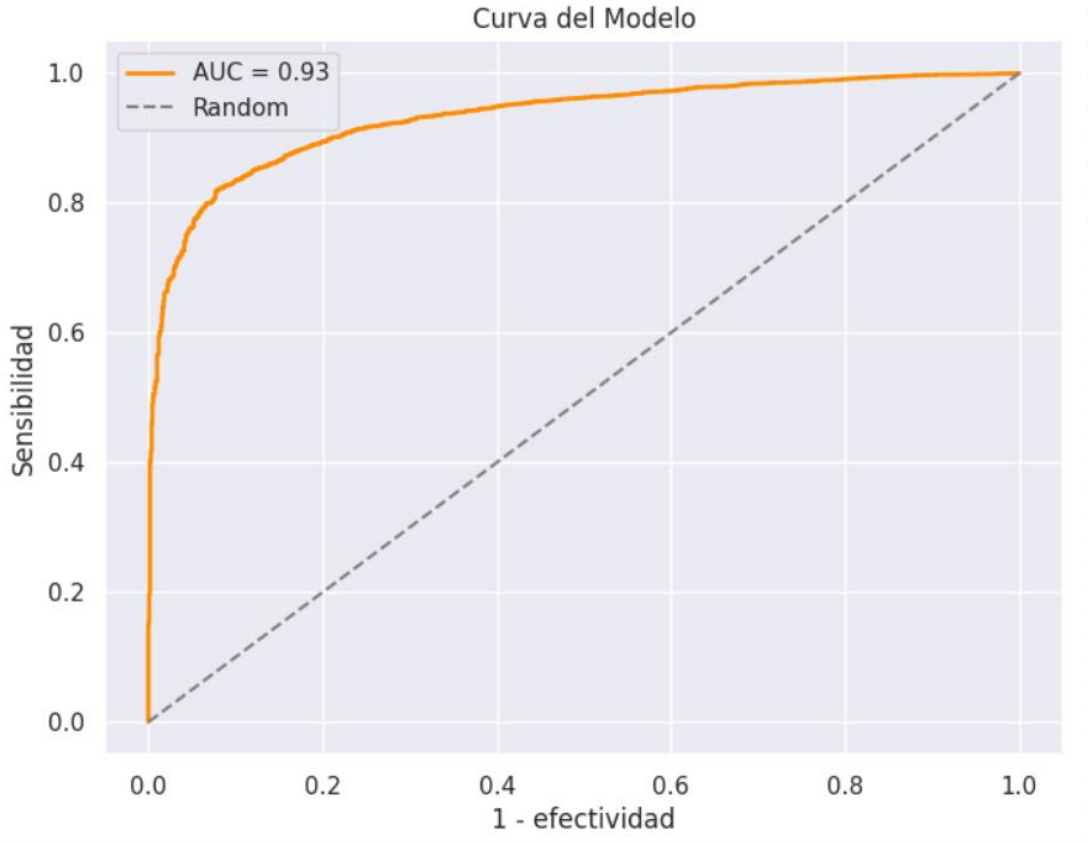


Red neuronal


```python
#@title **RED NEURONAL**
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

a = pd.read_csv("datamod.csv", sep=",")


X = a.drop('y', axis=1)
y = a['y']
label_encoder = LabelEncoder()
for col in ['job', 'marital', 'education', 'default', 'housing', 'loan', 'month', 'poutcome']:
    X[col] = label_encoder.fit_transform(X[col])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)


def build_model():
    model = Sequential()
    model.add(Dense(128, input_dim=X_train.shape[1], activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    optimizer = Adam(learning_rate=0.001)
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model

model = build_model()
model.fit(X_train, y_train, epochs=30, batch_size=64, validation_split=0.2)
```

Red neuronal

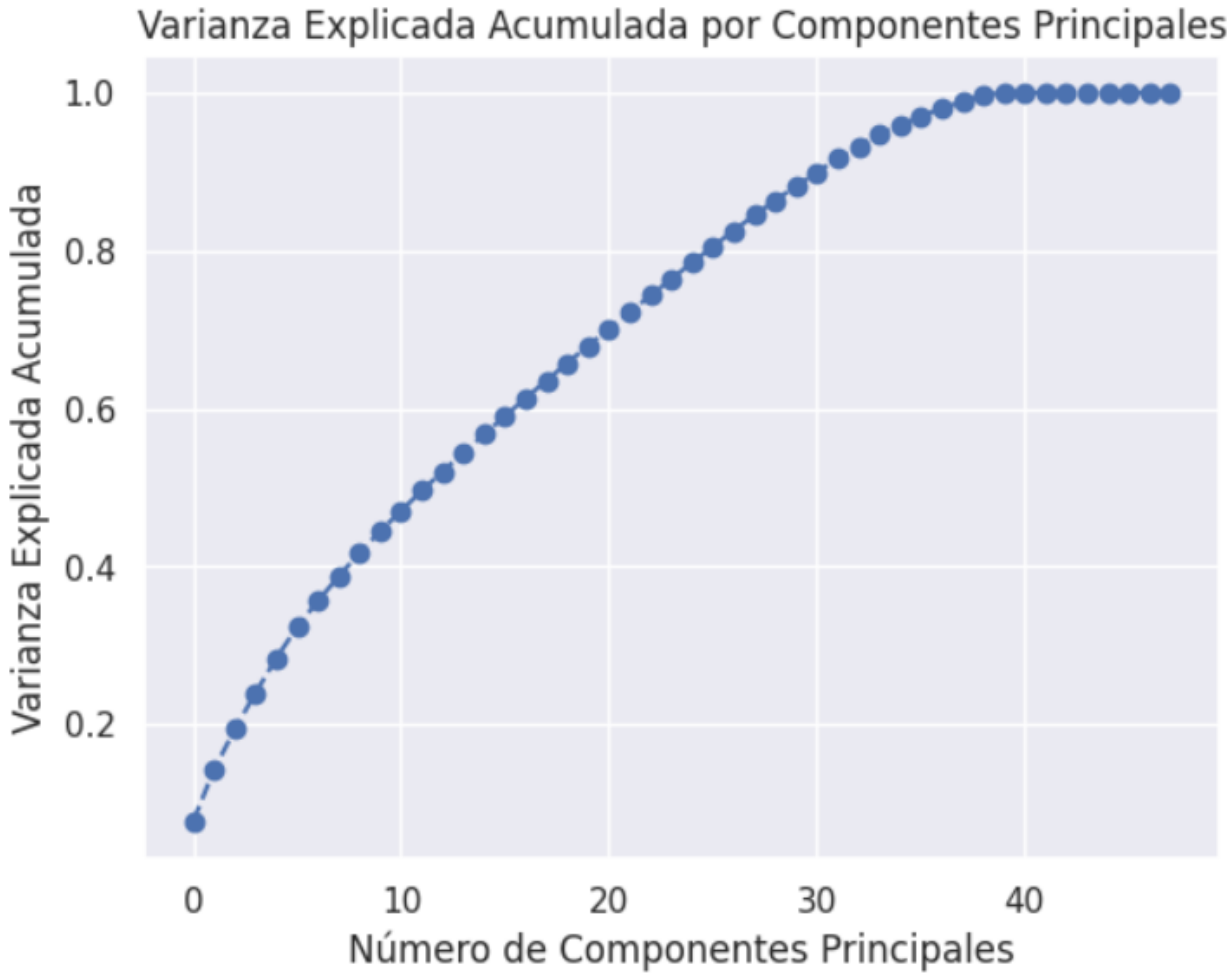
```
Epoch 1/30
453/453 [==============================] - 6s 6ms/step - loss: 0.2787 - accuracy: 0.8841 - val_loss: 0.2172 - val_accuracy: 0.9031

Epoch 30/30
453/453 [==============================] - 2s 5ms/step - loss: 0.1971 - accuracy: 0.9098 - val_loss: 0.1973 - val_accuracy: 0.9108
```



Varianza explicada - PCA


```python
#@title **VARIANZA EXPLICADA**
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

X_encoded = pd.get_dummies(X, columns=['job', 'marital', 'education', 'default', 'housing', 'loan', 'month', 'poutcome'])
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_encoded)

eigenvalues, eigenvectors = np.linalg.eig(np.cov(X_scaled.T))

order_of_importance = np.argsort(eigenvalues)[::-1]

sorted_eigenvalues = eigenvalues[order_of_importance]
sorted_eigenvectors = eigenvectors[:, order_of_importance]

explained_variance_ratio = sorted_eigenvalues / sum(sorted_eigenvalues)
explained_variance_cumulative = np.cumsum(explained_variance_ratio)

print("Explained Variance for each Principal Component:")
print(explained_variance_ratio)

print("\nCumulative Explained Variance:")
print(explained_variance_cumulative)
```

Varianza explicada - PCA



PCA

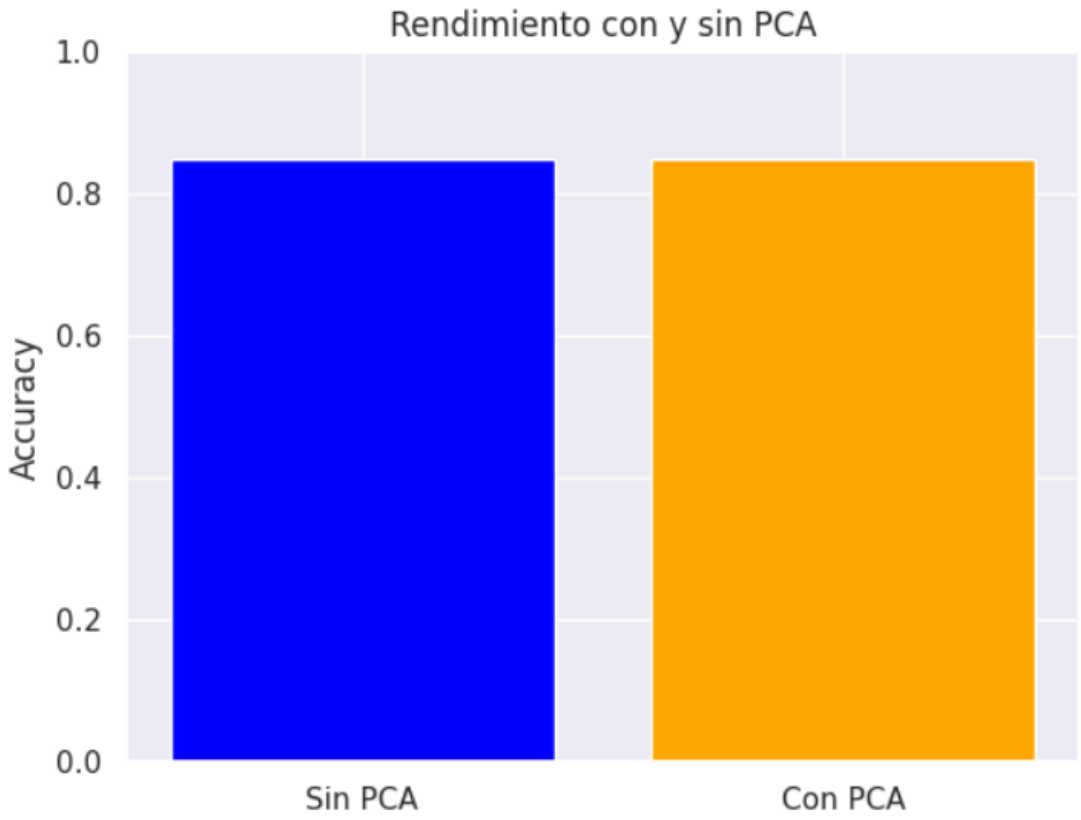

```python
#@title  **CON Y SIN PCA**
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

X_encoded = pd.get_dummies(X, columns=['job', 'marital', 'education', 'default', 'housing', 'loan', 'month', 'poutcome'])

X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y, test_size=0.7, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

est_without_pca = GaussianNB()
est_without_pca.fit(X_train_scaled, y_train)
y_pred_without_pca = est_without_pca.predict(X_test_scaled)

print("Entrenamiento y evaluación sin PCA:")
print("Accuracy sin PCA:", accuracy_score(y_test, y_pred_without_pca))
```

PCA


```
24 Components = 35
25 pca = PCA(n_components=Components, whiten=True)
26 pca.fit(X_train_scaled)
27
28 X_pca_train = pca.transform(X_train_scaled)
29 X_pca_test = pca.transform(X_test_scaled)
30
31 est_with_pca = GaussianNB()
32 est_with_pca.fit(X_pca_train, y_train)
33 y_pred_with_pca = est_with_pca.predict(X_pca_test)
34
35 print("\nEntrenamiento y evaluación con PCA:")
36 print(f"Número de componentes principales utilizadas: {Components}")
37 print("Accuracy con PCA:", accuracy_score(y_test, y_pred_with_pca))
```

PCA

```
Entrenamiento y evaluación sin PCA:
Accuracy sin PCA: 0.8490899898887766

Entrenamiento y evaluación con PCA:
Número de componentes principales utilizadas: 35
Accuracy con PCA: 0.8501959049544995
```



Universidad Industrial de Santander

Aniversario UIS 1948 - 2023

Legado **académico** y **cultural** de los **santandereanos**

¡Gracias!

#LaUISqueQueremos