

Problema 2

En el último mes, han llegado muchos más registros de los que puede atender el equipo de crédito. Alguien sugirió crear un modelo para poder decidir a quien prestarle

¿Crees que sea una buena idea? ¿ Por qué?

R//: Sí es una buena idea. Hay varias razones por las que es una buena idea crear un modelo de IA para automatizar procesos:

- Mejora de la eficiencia: La automatización de procesos con IA puede ayudar a reducir el tiempo y los recursos necesarios para completar tareas repetitivas y tediosas. Esto liberaría la carga del equipo y permitiría que se centraran en temas mas importantes y de mayor valor.
- Reducción de errores: Al eliminar la intervención humana en procesos repetitivos, se reduce la posibilidad de errores y se mejora la precisión de los resultados.
- Ahorro de costos: La automatización de procesos puede ayudar a reducir los costos a largo plazo.
- Mejora de la toma de decisiones: Los modelos de IA pueden analizar grandes cantidades de datos en tiempo real y tomar decisiones, lo que puede mejorar la eficiencia y la efectividad en la toma de decisiones.

En resumen, la automatización de procesos con IA puede mejorar la eficiencia, reducir los errores, ahorrar costos y mejorar la toma de decisiones, lo que puede ser una gran ventaja para cualquier organización.


Si la respuesta fue sí, ¿Qué tipo de problema es según los datos que tienes? (Supervisado o No Supervisado)

R//: Es un problema supervisado de clasificación. La sugerencia de crear un modelo para decidir a quién prestarle sugiere que existen datos etiquetados previamente que se pueden utilizar para entrenar el modelo. Es decir, que se tienen registros con información sobre los solicitantes de crédito y se sabe qué solicitantes han sido aprobados o rechazados en el pasado.

Haz las transformaciones que necesites a los datos y desarrolla algún modelo.

```
#Importar librerias
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OrdinalEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import joblib
```

```
# Cargar el dataset
data_url = "https://raw.githubusercontent.com/resuelve/resuelve-ia-prueba/master/datos_prestamo.csv"
df = pd.read_csv(data_url)
df
```

	Unnamed: 0 int64 0 - 556 	Fecha_registro o... 02/27/2019 ... 0.9% 02/26/2019 ... 0.7% 414 others 98.4%	Fecha_contacto o... 02/28/2019 ... 0.7% 02/28/2019 ... 0.5% 436 others 98.7%	Id object LP002894 0.4% LP002424 0.4% 535 others 99.3%	Genero object Hombre 80.1% Mujer 17.8% Missing 2.2%	Casado object Si 66.1% No 33.4% Missing 0.5%	Dependientes obj... 0 56.7% 3 others 40.9% Missing 2.3%	Educacion obje... Graduado 7... No Graduado . 2...
510	510	02/27/2019 07:24 AM	02/28/2019 06:31 PM	LP001322	Hombre	No	0	Graduado
511	511	02/28/2019 08:59 AM	03/01/2019 04:08 PM	LP002734	Hombre	Si	0	Graduado
512	512	02/26/2019 09:14 AM	02/27/2019 02:57 PM	LP001616	Hombre	Si	1	Graduado
513	513	02/28/2019 09:48 AM	03/01/2019 05:24 PM	LP002180	Hombre	No	0	Graduado
514	514	02/28/2019 10:25 AM	03/01/2019 01:17 PM	LP002862	Hombre	Si	2	No Graduado
515	515	02/26/2019 07:21 AM	02/27/2019 01:14 PM	LP002100	Hombre	No	nan	Graduado
516	516	02/27/2019 11:37 AM	02/28/2019 06:55 PM	LP002348	Hombre	Si	0	Graduado
517	517	02/27/2019 10:21 AM	02/28/2019 01:21 PM	LP002784	Hombre	Si	1	No Graduado

518	518	02/26/2019 10:17 AM	02/27/2019 03:07 PM	LP002794	Mujer	No	0	Graduado
519	519	02/26/2019 10:46 AM	02/27/2019 06:49 PM	LP002369	Hombre	Si	0	Graduado

```
# Seleccionando features(X) y target(y)
X = df.drop("Estatus_prestamo", axis=1)
y = df["Estatus_prestamo"]
```

```
# Dividir en train y test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Definir columnas numerias(num_cols) y categoricas(cat_cols)
#Podemos ignorar las variables Fecha_registro, Fecha_contacto, Asesor_asignado y Id ya que no aportan información relevante para el problema
num_cols = ['Salario', 'Salario_Pareja', 'Credito_pedido', 'Plazo_prestamo', 'Historial_crediticio']
cat_cols = ['Genero', 'Casado', 'Dependientes', 'Educacion', 'Trabaja_para_el', 'Area_vivienda']
```

```
# Se crean numeric y categoric pipelines
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean'))
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OrdinalEncoder())
])

# Se combinan los pipelines con ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, num_cols),
        ('cat', categorical_transformer, cat_cols)
    ])

```

```
# Se crean los pipelines con los clasificadores
''' Se usa el preprocessor como el primer paso del pipeline,
    luego se escala los datos numericos y luego se aplica uno de los clasificadores
    (RandomForest, Vector Classifier (SVC), Logistic)'''
pipeline_rf = Pipeline(steps=[('preprocessor', preprocessor),
                              ('scaler', StandardScaler()),
                              ('classifier', RandomForestClassifier())])

pipeline_svc = Pipeline(steps=[('preprocessor', preprocessor),
                              ('scaler', StandardScaler()),
                              ('classifier', SVC())])

pipeline_lr = Pipeline(steps=[('preprocessor', preprocessor),
                              ('scaler', StandardScaler()),
                              ('classifier', LogisticRegression())])

# Se evalua el performance de cada pipeline usando corss-validation para estimar desempeño cada modelo
scores_rf = cross_val_score(pipeline_rf, X_train, y_train, cv=5)
scores_svc = cross_val_score(pipeline_svc, X_train, y_train, cv=5)
scores_lr = cross_val_score(pipeline_lr, X_train, y_train, cv=5)
```

```
print('Random Forest Classifier Cross-Validation Puntajes:', scores_rf)
print('Support Vector Classifier Cross-Validation Puntajes:', scores_svc)
print('Logistic Regression Cross-Validation Puntajes:', scores_lr)
```

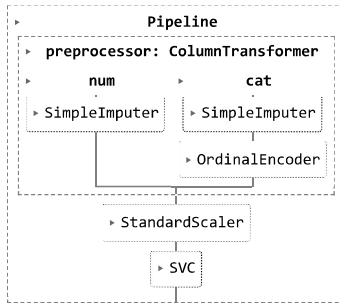
```
Random Forest Classifier Cross-Validation Puntajes: [0.8988764 0.87640449 0.76404494 0.85393258 0.86516854]
Support Vector Classifier Cross-Validation Puntajes: [0.91011236 0.88764045 0.78651685 0.86516854 0.8988764 ]
Logistic Regression Cross-Validation Puntajes: [0.91011236 0.87640449 0.76404494 0.87640449 0.91011236]
```

```
# Media y desviacion estandar de los puntajes de cross-validation para cada modelo
print('Random Forest Cross-Validation Puntaje: Mean - %.7g | Std - %.7g' % (np.mean(scores_rf), np.std(scores_rf)))
print('Support Vector Classifier Cross-Validation Puntaje: Mean - %.7g | Std - %.7g' % (np.mean(scores_svc), np.std(scores_svc)))
print('Logistic Regression Cross-Validation Puntaje: Mean - %.7g | Std - %.7g' % (np.mean(scores_lr), np.std(scores_lr)))

# Seleccion del mejor modelo basado en la media del puntaje del cross-validation
if np.mean(scores_rf) > np.mean(scores_svc) and np.mean(scores_rf) > np.mean(scores_lr):
    best_model = pipeline_rf
elif np.mean(scores_svc) > np.mean(scores_lr):
    best_model = pipeline_svc
```

```
else:
    best_model = pipeline_lr
best_model.fit(X_train,y_train)
```

Random Forest Cross-Validation Puntaje: Mean - 0.8516854 | Std - 0.0462725
 Support Vector Classifier Cross-Validation Puntaje: Mean - 0.8696629 | Std - 0.0441503
 Logistic Regression Cross-Validation Puntaje: Mean - 0.8674157 | Std - 0.05383887



```
# Guardar el modelo
filename = 'best_model.pkl'
joblib.dump(best_model, filename)
```

```
['best_model.pkl']
```

¿Cómo sabes que es un buen modelo?

Para saber si es un buen modelo se tiene que probar con nuevos datos y hacer sus predicciones.

En este caso se utilizarán dos métricas (accuracy_score y confusion_matrix) para determinar si es un buen modelo:

Accuracy_score: Es una medida de cuán preciso es el modelo al predecir las etiquetas de las muestras de datos de prueba.

Confusion_matrix: Con la matriz de confusión, puede evaluar el rendimiento del modelo en términos de sensibilidad (precisión en la identificación de la clase positiva), especificidad (precisión en la identificación de la clase negativa), muestra las predicciones correctas y las predicciones incorrectas del modelo en términos de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

```
# Cargar el modelo
best_model = joblib.load('/work/best_model.pkl')
```

```
# Usar el modelo para hacer predicciones con nueva data
y_predictions = best_model.predict(X_test)

# Accuracy score
accuracy = accuracy_score(y_test, y_predictions)
print("Accuracy Score:", accuracy)
```

```
Accuracy Score: 0.8482142857142857
```

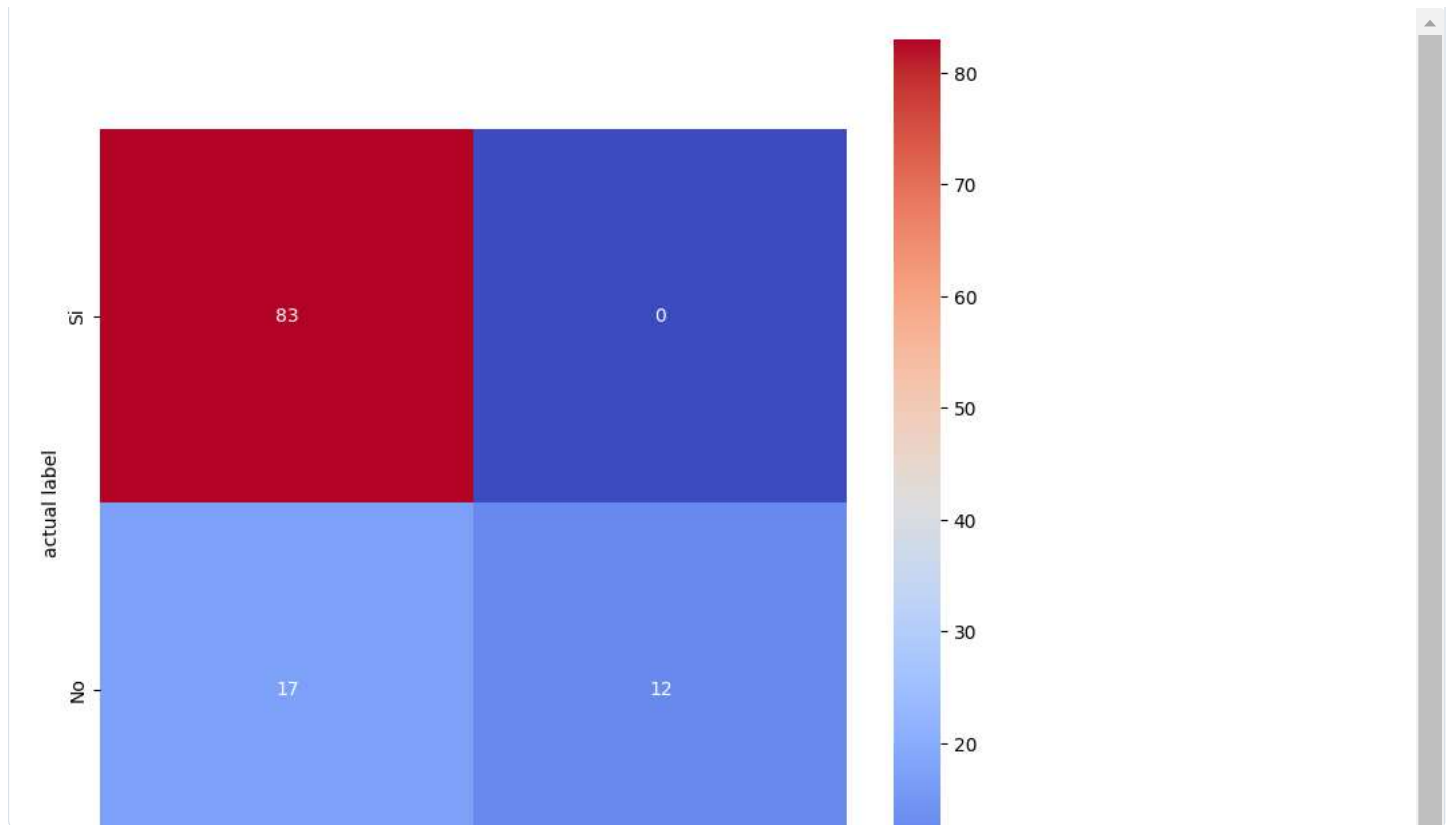
La métrica de accuracy score es útil para evaluar el rendimiento del modelo en general, ya que mide la proporción de aciertos sobre el total de predicciones.

```
# Confusion matrix
confusion_mat = confusion_matrix(y_test, y_predictions, labels=['Si', 'No'])
print('Matriz de confusion:\n', confusion_mat)
```

```
Matriz de confusion:
[[83  0]
 [17 12]]
```

```
plt.figure(figsize=(9,9))
sns.heatmap(confusion_mat, annot=True, square=True, cmap='coolwarm')
plt.xticks(ticks=[0.5, 1.5], labels=['Si', 'No'])
plt.yticks(ticks=[0.5, 1.5], labels=['Si', 'No'])
plt.ylabel('actual label')
plt.xlabel('predicted label')
```

```
Text(0.5, 137.2222222222217, 'predicted label')
```



No se puede determinar si el modelo es bueno o malo basándose únicamente en los valores de la matriz de confusión sin tener en cuenta el contexto y las métricas relevantes para el problema en particular. La matriz de confusión muestra el número de predicciones correctas y incorrectas, pero no da una visión completa de la eficacia del modelo.

Por ejemplo, si el objetivo de la empresa es minimizar el número de falsos negativos (personas que no se les dio el crédito pero el modelo predijo que si), entonces una matriz de confusión con un bajo número de falsos negativos sería buena. Pero si el objetivo de la empresa es minimizar el número de falsos positivos (personas que si se les dio el crédito pero el modelo predijo que no), entonces una matriz de confusión con un bajo número de falsos positivos sería buena.

```
print(f'En este caso, el número de falsos negativos es {confusion_mat[1][0]} y el número de falsos positivos es {confusion_mat[0][1]}.'
```

En este caso, el número de falsos negativos es 17 y el número de falsos positivos es 0.

Dependiendo de los objetivos de la empresa, es posible que una de estas métricas sea más importante que la otra. Por lo tanto, es importante evaluar cuidadosamente los resultados en relación con los objetivos y la política de la empresa antes de decidir si los resultados son buenos o malos.