Calibrar modelos de machine learning

Joaquín Amat Rodrigo

Octubre, 2020

Introducción

Muchos de modelos de *machine learning* empleados en problemas de clasificación (regresión logística, random forest (https://www.cienciadedatos.net/documentos/py08_random_forest_python), gradient boosting (https://www.cienciadedatos.net/documentos/py09_gradient_boosting_python),...) son capaces de estimar un valor de probabilidad asociado a cada predicción. ¿Hasta qué punto se debe de confiar en estas probabilidades? Aquí es donde entra en juego el concepto de calibración.

Un modelo calibrado es aquel en el que, el valor estimado de probabilidad, puede interpretarse directamente como la confianza que se tiene de que la clasificación predicha es correcta. Por ejemplo, si para un modelo de clasificación binaria (perfectamente calibrado) se seleccionan las predicciones cuya probabilidad estimada es de 0.8, en torno al 80% estarán bien clasificadas.

Además de la calibración, a la hora de interpretar las probabilidades es muy importante tener en cuenta la diferencia entre la "visión" que tiene el modelo del mundo y el mundo real. Todo lo que sabe un modelo es lo que ha podido aprender de los datos de entrenamiento y, por lo tanto, tiene una "visión" limitada. Por ejemplo, supóngase un problema de clasificación en el que se predice la localización de una casa (ciudad o rural) en función de sus características arquitectónicas, y que, en los datos de entrenamiento, todas las casas que tienen chimenea están en zona urbana independientemente del valor que tomen el resto de predictores. Cuando el modelo trate de predecir una nueva observación, si tiene chimenea, clasificará a la casa como zona urbana con un 100% de probabilidad. Sin embargo, esto no significa que sea inequívocamente cierto, podría haber casas en zonas de ciudad que sí tengan chimenea pero, al no estar presentes en los datos de entrenamiento, el modelo no contempla esta posibilidad.

Teniendo en cuenta todo esto, hay que considerar las probabilidades generadas por el modelo como la seguridad que tiene este, desde su visión limitada, al realizar las predicciones. Pero no como la probabilidad en el mundo real de que así lo sea.

A lo largo de este documento, se describe cómo cuantificar el grado de calibración de un modelo **Scikit-learn** de **Python** y cómo corregirlo.

Calibración

Definición matemática

La calibración de un modelo de clasificación consiste en reajustar las probabilidades predichas para que correspondan con la proporción de casos reales observados. En otras palabras, corregir las probabilidades predichas por un modelo cuando este las subestima o sobrestima.

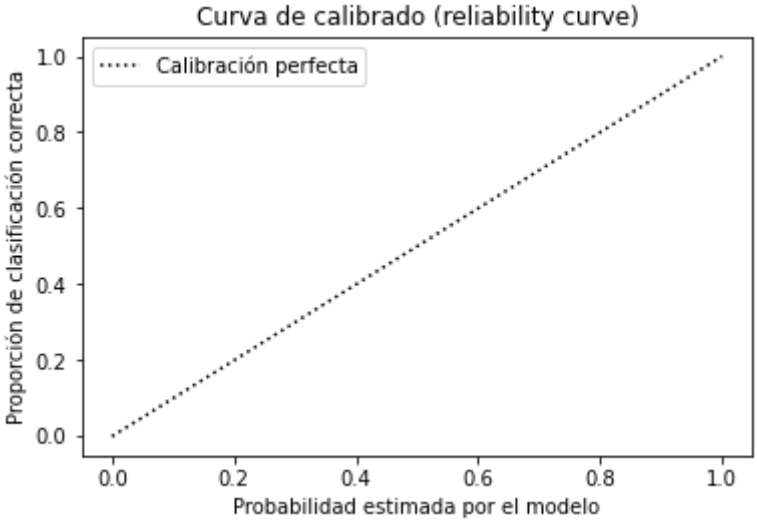
Un modelo está perfectamente calibrado cuando, para cualquier valor p, la clasificación predicha con una confianza (probabilidad) de p es correcta el $100 * p$ por ciento de las veces.

$$P(\hat{Y} = Y|\hat{P} = p) = p$$

$$p \in [0, 1]$$

Por ejemplo, si se seleccionan las observaciones cuya probabilidad predicha es $\hat{p} = 0.8$, es de esperar que el porcentaje de esas observaciones bien clasificadas sea del 80%.

Esto puede generalizarse para todo el rango de probabilidades [0,1], lo que da lugar a una diagonal perfecta.



Calibración no implica *accuracy*

Una buena calibración no implica que el modelo sea bueno clasificando (*accuracy* elevado). Supóngase un modelo que clasifica aleatoriamente las observaciones en cada grupo con un 50% de probabilidad. Este modelo tendrá solo un 50% de *accuracy* pero un calibrado perfecto. Lo mismo puede ocurrir al contrario, un modelo con muy buena capacidad de clasificación pero que siempre sobrestime las probabilidades predichas.

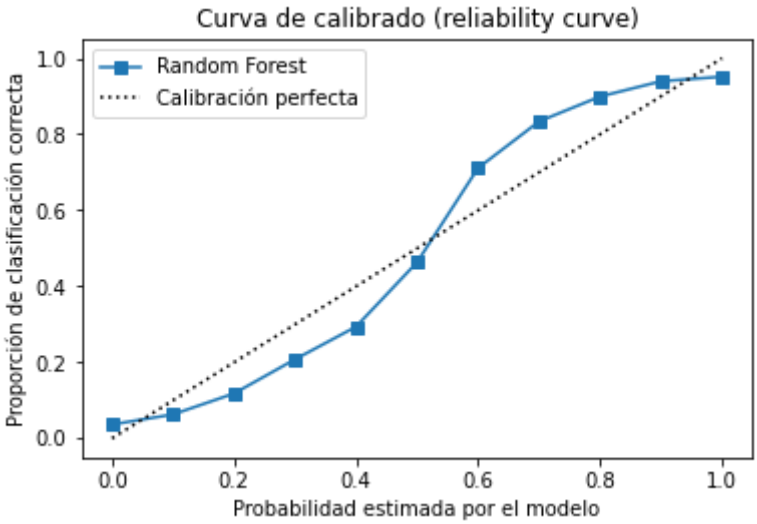
El proceso de calibración es importante cuando se quieren emplear las probabilidades asociadas a las predicciones. Si únicamente son de interés las clasificaciones finales, la calibración no aporta valor. Esto último es importante tenerlo en cuenta ya que, calibrar un modelo, puede implicar reducir su porcentaje de clasificaciones correctas (*accuracy*).

Curvas de calibrado

El proceso más empleado para saber si un modelo está bien calibrado es generar la curva de calibrado o *reliability plots*. La forma de calcularla es:

- Se ordenan todas las predicciones del modelo de menor a mayor probabilidad y se agrupan en intervalos (*bins*).
- Se calcula la proporción de clasificaciones correctas en cada *bin* (proporción empírica).
- Se calcula la confianza del *bin* como el valor promedio de las probabilidades estimadas por el modelo para todas las observaciones que forman parte del *bin* (confianza del modelo).

Cuanto mejor calibrado esté el modelo, más próximos serán los valores de proporción empírica y de confianza, es decir, más se aproxima la curva obtenida a la diagonal. La curva de calibrado queda por encima de la diagonal si el modelo tiende a infravalorar las probabilidades y por debajo si las sobrevalora. Véase la siguiente imagen.



La gráfica muestra la curva de calibrado de un modelo *random forest*. El patrón de la curva indica que, para valores bajos, el modelo tiende a sobrevalorar sus probabilidades y, para valores altos, infravalorarlas. Por ejemplo, para una probabilidad estimada por el modelo de 0.4, solo entorno al 2% de las observaciones están bien clasificadas.

Brier score

Aunque las curvas de calibración aportan información detallada, es interesante disponer de una métrica que permita cuantificar con un único valor la calidad de calibración del modelo. *Brier score* es la diferencia cuadrática media (*mean squared difference*) entre la probabilidad estimada por el modelo y la probabilidad real (1 para la clase positiva y 0 para la negativa). Cuanto menor es su valor, mejor calibrado está el modelo. Esta métrica es adecuada solo para clasificaciones binarias.

Calibrar un modelo

Calibrar un modelo consiste en corregir las desviaciones de su curva de calibrado respecto a la diagonal (calibración perfecta). Esto puede conseguirse con un segundo modelo que aprenda como hacer la corrección. Los modelos empleados para esta corrección suelen ser regresión logística (*platt scaling*) o regresión isotónica. La regresión isotónica solo se recomienda cuando se dispone de >1000 observaciones, por su facilidad de *overfitting*.

El entrenamiento del modelo correctivo no debe hacerse con los mismos datos con los que se ha entrenado el modelo principal para evitar así el *overfitting*. En su lugar, se debe utilizar un conjunto de validación o validación cruzada.

Ejemplo

Librerías

Las librerías utilizadas en este documento son:

In [155]:
```
# Tratamiento de datos
# ==============================================================================
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification

# Gráficos
# ==============================================================================
import matplotlib.pyplot as plt

# Preprocesado y modelado
# ==============================================================================
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.calibration import calibration_curve
from sklearn.calibration import CalibratedClassifierCV
from sklearn.metrics import brier_score_loss
from sklearn.metrics import classification_report

# Configuración warnings
# ==============================================================================
import warnings
warnings.filterwarnings('once')
warnings.simplefilter('ignore', (DeprecationWarning))
```

Datos

Para este ejemplo se simulan datos empleando la función `make_classification` de **scikitlearn**.

In [156]:
```
# Datos simulados (clasificación binaria)
# ==============================================================
================
X, y = make_classification(
            n_samples     = 10000,
            n_classes     = 2,
            n_features    = 20,
            n_informative = 2,
            n_redundant   = 2,
            random_state  = 42
)
```

In [157]:
```
# Número de observaciones por clase
# ==============================================================
================
unique, counts = np.unique(y, return_counts=True)
dict(zip(unique, counts))
```

Out[157]:
```
{0: 5005, 1: 4995}
```

Modelo clasificación sin calibrar

Modelo

In [158]:
```
# División de los datos en train y test
# ==============================================================
================
X_train, X_test, y_train, y_test = train_test_split(
                                        X,
                                        y,
                                        test_size    = 0.5,
                                        random_state = 123
                                    )
# Creación del modelo
# ==============================================================
================
modelo = RandomForestClassifier(
            n_estimators = 10,
            max_features = 'auto',
            random_state = 123
         )

#modelo = LogisticRegression()
# Entrenamiento del modelo
# ==============================================================
================
modelo.fit(X_train, y_train)
```

Out[158]:
```
RandomForestClassifier(n_estimators=10, random_state=123)
```

In [159]:
```
# Predicción con probabilidades
# ==============================================================
predicciones  = modelo.predict_proba(X = X_test)
# Se extraen las probabilidades de la clase positiva
prob_positivo = predicciones[:, 1]
```

Curva calibrado

La curva de calibrado, para modelos de clasificación binaria, puede calcularse empleando la función `calibration_curve()` del módulo `sklearn.calibration`.

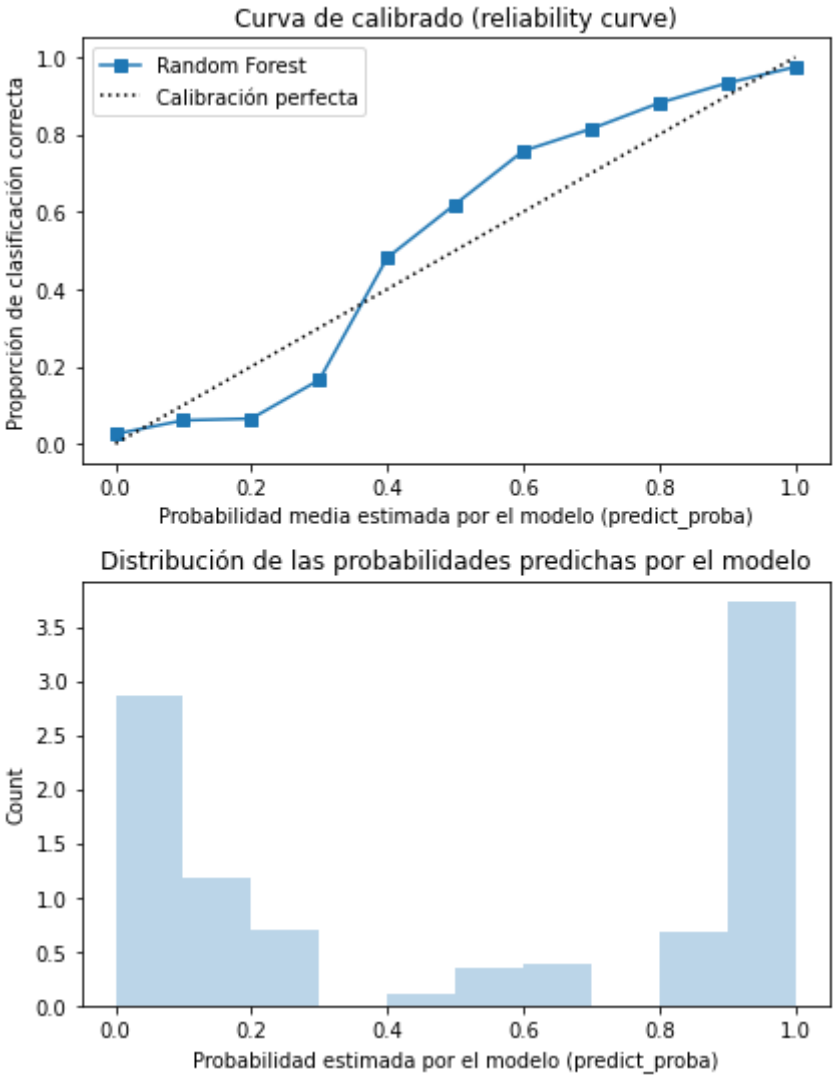
In [160]:
```
fraccion_positivos, media_prob_predicha = calibration_curve(y_t
est, prob_positivo, n_bins=20)
```

In [161]:
```
fig, axs = plt.subplots(nrows=2, ncols=1,figsize=(6, 2*3.84))

axs[0].plot(media_prob_predicha, fraccion_positivos, "s-", label="Random Forest")
axs[0].plot([0, 1], [0, 1], "k:", label="Calibración perfecta")
axs[0].set_ylabel("Proporción de clasificación correcta")
axs[0].set_xlabel("Probabilidad media estimada por el modelo (predict_proba)")
axs[0].set_title('Curva de calibrado (reliability curve)')
axs[0].legend()

axs[1].hist(prob_positivo, range=(0, 1), bins=10, density=True, lw=2, alpha = 0.3)
axs[1].set_xlabel("Probabilidad estimada por el modelo (predict_proba)")
axs[1].set_ylabel("Count")
axs[1].set_title('Distribución de las probabilidades predichas por el modelo')

plt.tight_layout()
plt.show();
```



El modelo generado no estás calibrado, tiende a sobrevalorar sus probabilidades cuando estas son inferiores a 0.4, y a infravalorarlas cundo son superiores a 0.4. Por ejemplo, para una probabilidad estimada por el modelo de 0.6, en torno a un 80% de las observaciones pertenecen a la clase positiva.

Métricas

In [162]:
```
brier_score = brier_score_loss(y_test, modelo.predict_proba(X = X_test)[:, 1])
print(f"Brier score = {brier_score}")
print("")
print(classification_report(y_test, modelo.predict(X = X_test)))
```

```
Brier score = 0.06009400000000001

              precision    recall  f1-score   support

           0       0.93      0.93      0.93      2501
           1       0.93      0.93      0.93      2499

    accuracy                           0.93      5000
   macro avg       0.93      0.93      0.93      5000
weighted avg       0.93      0.93      0.93      5000
```

Modelo clasificación calibrado

La calibración de modelos en **scikitlearn** esta implementada de forma que, el objeto final, contiene tanto el modelo principal como el modelo de calibración. Esto permite que al aplicar el método `.predic()` los resultados ya estén corregidos.

Para crear un modelo calibrado se emplea la clase `CalibratedClassifierCV`, cuyos argumentos son:

- `base_estimator`: el modelo que se quiere calibrar.
- `method`: el tipo de modelo empleado para la calibración ('sigmoid' o 'isotonic').
- `cv`: estrategia de validación para el entrenamiento de ambos modelos. Si se indica 'prefit', se asume que el `base_estimator` ya está entrenado y todos los datos se utilizan para el modelo de calibrado.

Modelo

In [163]:
```
# Creación del modelo base
# ==============================================================
modelo = RandomForestClassifier(
            n_estimators = 10,
            max_features = 'auto',
            random_state = 123
         )

# Creación del modelo calibrado
# ==============================================================
modelo_calibrado = CalibratedClassifierCV(modelo, cv=3, method
='isotonic')

# Entrenamiento del modelo base y d ela calibración
# ==============================================================
_ = modelo_calibrado.fit(X_train, y_train)
```

In [164]:
```
# Predicción con probabilidades calibradas
# ==============================================================
predicciones  = modelo_calibrado.predict_proba(X = X_test)
# Se extraen las probabilidades de la clase positiva
prob_positivo = predicciones[:, 1]
```

Curva calibrado

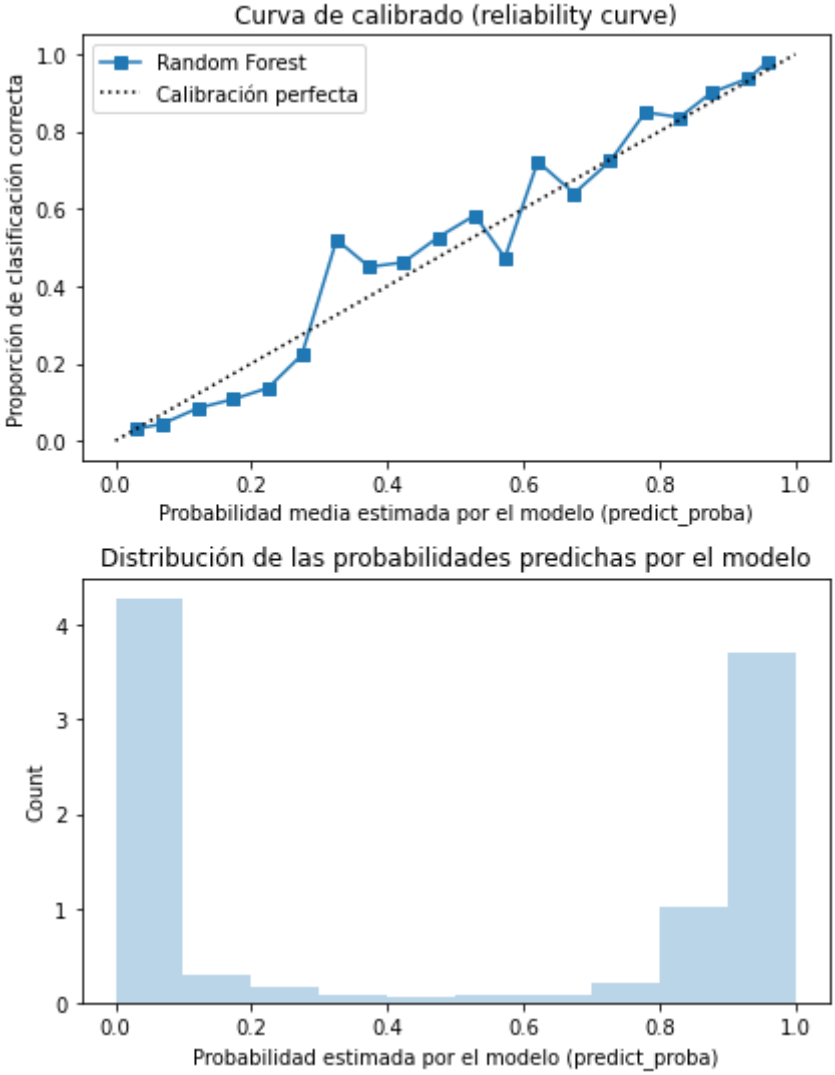
In [165]:
```
fraccion_positivos, media_prob_predicha = calibration_curve(y_t
est, prob_positivo, n_bins=20)
```

In [166]:
```
fig, axs = plt.subplots(nrows=2, ncols=1,figsize=(6, 2*3.84))

axs[0].plot(media_prob_predicha, fraccion_positivos, "s-", label="Random Forest")
axs[0].plot([0, 1], [0, 1], "k:", label="Calibración perfecta")
axs[0].set_ylabel("Proporción de clasificación correcta")
axs[0].set_xlabel("Probabilidad media estimada por el modelo (predict_proba)")
axs[0].set_title('Curva de calibrado (reliability curve)')
axs[0].legend()

axs[1].hist(prob_positivo, range=(0, 1), bins=10, density=True, lw=2, alpha = 0.3)
axs[1].set_xlabel("Probabilidad estimada por el modelo (predict_proba)")
axs[1].set_ylabel("Count")
axs[1].set_title('Distribución de las probabilidades predichas por el modelo')

plt.tight_layout()
plt.show();
```



Tras la calibración, las probabilidades generadas se aproximan más a la diagonal.

Métricas

In [167]:
```
brier_score = brier_score_loss(y_test, modelo_calibrado.predict
_proba(X = X_test)[:, 1])
print(f"Brier score = {brier_score}")
print("")
print(classification_report(y_test, modelo_calibrado.predict(X
= X_test)))
```

```
Brier score = 0.05578011672300572

              precision    recall  f1-score   support

           0       0.94      0.92      0.93      2501
           1       0.92      0.94      0.93      2499

    accuracy                           0.93      5000
   macro avg       0.93      0.93      0.93      5000
weighted avg       0.93      0.93      0.93      5000
```

Se ha conseguido reducir el *brier score*, indicativo de que ha mejorado la calibración del modelo. El *accuracy* del modelo no se ha visto perjudicado.

Información de sesión

In [168]:
```
from sinfo import sinfo
sinfo()
```

```
-----
matplotlib  3.3.2
numpy       1.19.2
pandas      1.1.3
sinfo       0.3.1
sklearn     0.23.2
-----
IPython             7.18.1
jupyter_client      6.1.2
jupyter_core        4.6.3
jupyterlab          2.1.3
notebook            6.0.3
-----
Python 3.8.6 (default, Oct 10 2020, 07:54:55) [GCC 5.4.0 20160609]
Linux-5.4.0-1028-aws-x86_64-with-glibc2.2.5
8 logical CPU cores, x86_64
-----
Session information updated at 2020-10-26 17:09
```

Bibliografía

sklearn.calibration (https://scikit-learn.org/stable/modules/calibration.html#calibration)

Applied ML 2020 - 10 - Calibration, Imbalanced data (https://www.youtube.com/watch?v=w3OPq0V8fr8#)

Model Calibration - is your model ready for the real world? - Inbar Naor - PyCon Israel 2018 (https://www.youtube.com/watch?v=FkfDlOnQVvQ)

Introduction to Machine Learning with Python: A Guide for Data Scientists libro (https://www.amazon.es/gp/product/1449369413/ref=as_li_qf_asin_il_tl?ie=UTF8&tag=cienciadedato-21&creative=24630&linkCode=as2&creativeASIN=1449369413&linkId=e07f892d9e2c458e

An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics) libro (https://www.amazon.es/gp/product/1461471370/ref=as_li_tl?ie=UTF8&camp=3638&creative=24630&creativeASIN=1461471370&linkCode=as2&tag=ci
21&linkId=64752a80078f4f017e81874b8cb355b7)

Applied Predictive Modeling by Max Kuhn and Kjell Johnson libro (https://www.amazon.es/gp/product/1461468485/ref=as_li_qf_asin_il_tl?ie=UTF8&tag=cienciadedato-21&creative=24630&linkCode=as2&creativeASIN=1461468485&linkId=f014c46d1f6c670f3

The Elements of Statistical Learning by T.Hastie, R.Tibshirani, J.Friedman libro (https://www.amazon.es/gp/product/B00M0R6ZJG/ref=as_li_qf_asin_il_tl?ie=UTF8&tag=cienciadedato-21&creative=24630&linkCode=as2&creativeASIN=B00M0R6ZJG&linkId=7d710cfdc19d2d7

¿Cómo citar este documento?

Calibrar probabilidades en modelos de machine learning por Joaquín Amat Rodrigo, disponible con licencia CC BY-NC-SA 4.0 en https://www.cienciadedatos.net/documentos/py11-calibrar-modelos-machine-learning.html

DOI 10.5281/zenodo.10006330
(https://doi.org/10.5281/zenodo.10006330)

¿Te ha gustado el artículo? Tu ayuda es importante

Mantener un sitio web tiene unos costes elevados, tu contribución me ayudará a seguir generando contenido divulgativo gratuito. ¡Muchísimas gracias! 😊