uc3m | Universidad Carlos III de Madrid

# Proyecto 3: Análisis de opinión

## Procesado del Lenguaje Natural

**Angel Navia Vázquez**

- 1.1 (January 2025) Revised and updated version

Departamento de Teoría de la Señal y Comunicaciones

**Universidad Carlos III de Madrid**

En este notebook debéis entrenar un clasificador de opinión utilizando los datos facilitados (**data_project_NLP_3_train.csv, data_project_NLP_3_val.csv**). En estos ficheros, las **opiniones positivas están codificadas como "1" y las negativas como "0"**.

Tenéis total libertad para elegir la mejor implementación. El modelo final será evaluado en un conjunto de test (no proporcionado).

Incluid aquí todas las pruebas de los diferentes modelos, así como la selección llevado a cabo. Se deberá guardar en ficheros el modelo elegido como ganador, para su uso posterior en el notebook "Proyecto_3_NLP_mejor_modelo.ipynb".

```
# Importar aquí las librerías necesarias
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.metrics import roc_auc_score
import joblib
```

```
# Vinculamos DRIVE para importar datos
from google.colab import drive
drive.mount("/content/drive")
MYDRIVE="/content/drive/MyDrive/2_CUATRI/NLP/LAB3/"
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
# Cargar los datos de entrenamiento y validación proporcionados
train_data = pd.read_csv(MYDRIVE+'data_project_NLP_3_train.csv')
val_data = pd.read_csv(MYDRIVE+'data_project_NLP_3_val.csv')

# Separamos en variables "X" e "y" cada dataset
X_train = train_data['texto']
y_train = train_data['opinion']

X_val = val_data['texto']
y_val = val_data['opinion']

# Aplicamos vectorización Tf-Idf al conjunto de entrenamiento
vectorizer = TfidfVectorizer(
    min_df=5,
    max_df=0.8,
    max_features=10000,
    ngram_range=(1, 2),
    sublinear_tf=True,
)

X_train_tfidf = vectorizer.fit_transform(X_train)
X_val_tfidf = vectorizer.transform(X_val)


# Entrenar los diferentes modelos
# Logistic Regression
clf_lr = LogisticRegression()
clf_lr.fit(X_train_tfidf, y_train)
y_pred_lr = clf_lr.predict(X_val_tfidf)

# SVM
clf_svm = SVC(probability=True)
clf_svm.fit(X_train_tfidf, y_train)
y_pred_svm = clf_svm.predict(X_val_tfidf)

# Multinomial Naive Bayes
clf_nb = MultinomialNB()
clf_nb.fit(X_train_tfidf, y_train)
y_pred_nb = clf_nb.predict(X_val_tfidf)

# Random Forest
clf_rf = RandomForestClassifier()
clf_rf.fit(X_train_tfidf, y_train)
y_pred_rf = clf_rf.predict(X_val_tfidf)


# elegir el modelo ganador
# Hallamos la AUC de cada modelo y comparamos
modelos = [clf_lr, clf_svm, clf_nb, clf_rf]
nombres_modelos = ['Logistic Regression', 'SVM', 'Multinomial Naive Bayes', 'Random For

for nombre, modelo in zip(nombres_modelos, modelos):

  if hasattr(modelo, 'predict_proba'):
    y_pred = modelo.predict_proba(X_val_tfidf)[:, 1]
```

```
    else:
      y_pred = modelo.predict(X_val_tfidf)

    auc = roc_auc_score(y_val, y_pred)
    print(f"* {nombre}: AUC = {auc}\n")
```

```
* Logistic Regression: AUC = 0.9022442872279435

* SVM: AUC = 0.9122805964355143

* Multinomial Naive Bayes: AUC = 0.8688216066299074

* Random Forest: AUC = 0.8874139419753049
```

```
# Almacenar en ficheros el modelo ganador para su uso posterior

# Guardamos el modelo SVM
joblib.dump(clf_svm, MYDRIVE + 'modelo_ganador.pkl')

# Guardamos el vectorizador usado
joblib.dump(vectorizer, MYDRIVE + 'vectorizador_modelo_ganador.pkl')
```

```
['/content/drive/MyDrive/2_CUATRI/NLP/LAB3/vectorizador_modelo_ganador.pkl']
```