In [122…
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import nltk
from nltk.tokenize import word_tokenize
from nltk import RegexpTokenizer
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

from wordcloud import WordCloud



from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
from sklearn.base import TransformerMixin, BaseEstimator


from sklearn.decomposition import TruncatedSVD, PCA
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.manifold import TSNE
```

In [123…
```python
raw_data = pd.read_excel('Train_textosODS.xlsx')
data = raw_data.copy()
data.head()
```

Out[123…

	textos	ODS
0	"Aprendizaje" y "educación" se consideran sinó...	4
1	No dejar clara la naturaleza de estos riesgos ...	6
2	Como resultado, un mayor y mejorado acceso al ...	13
3	Con el Congreso firmemente en control de la ju...	16
4	Luego, dos secciones finales analizan las impl...	5

In [124…
```python
x_train, x_test, y_train, y_test = train_test_split(data['textos'], data['ODS'], test_size=0.2, random_state=42)
```

In [125…
```python
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/I508111/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[125… True

In [126…
```python
bow_list =[
    {
        "vectorized": "count",
        "reduction": "svd",
        "train_data": None,
        "test_data": None
    },
    {
        "vectorized": "tfidf",
        "reduction": "svd",
        "train_data": None,
        "test_data": None
    },
    {
        "vectorized": "count",
        "reduction": "pca",
        "train_data": None,
        "test_data": None
    },
    {
        "vectorized": "tfidf",
        "reduction": "pca",
        "train_data": None,
        "test_data": None
    }
]
```

In [127…
```python
class TextPreprocessorTransformer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self
    def transform(self, X, y=None):
        return X.apply(self.text_preprocess)

    def text_preprocess(self, text):
        tokenizer = RegexpTokenizer(r'\w+')
        stemmer = PorterStemmer()
```

```python
        tokens = tokenizer.tokenize(text)
        tokens = [word for word in tokens if word not in stopwords.words('spanish')]
        tokens = [stemmer.stem(word) for word in tokens]
        return ' '.join(tokens)
```

In [128…

```python
class ToDenseTransformer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self
    def transform(self, X, y=None):
        return X.toarray()
```

In [129…

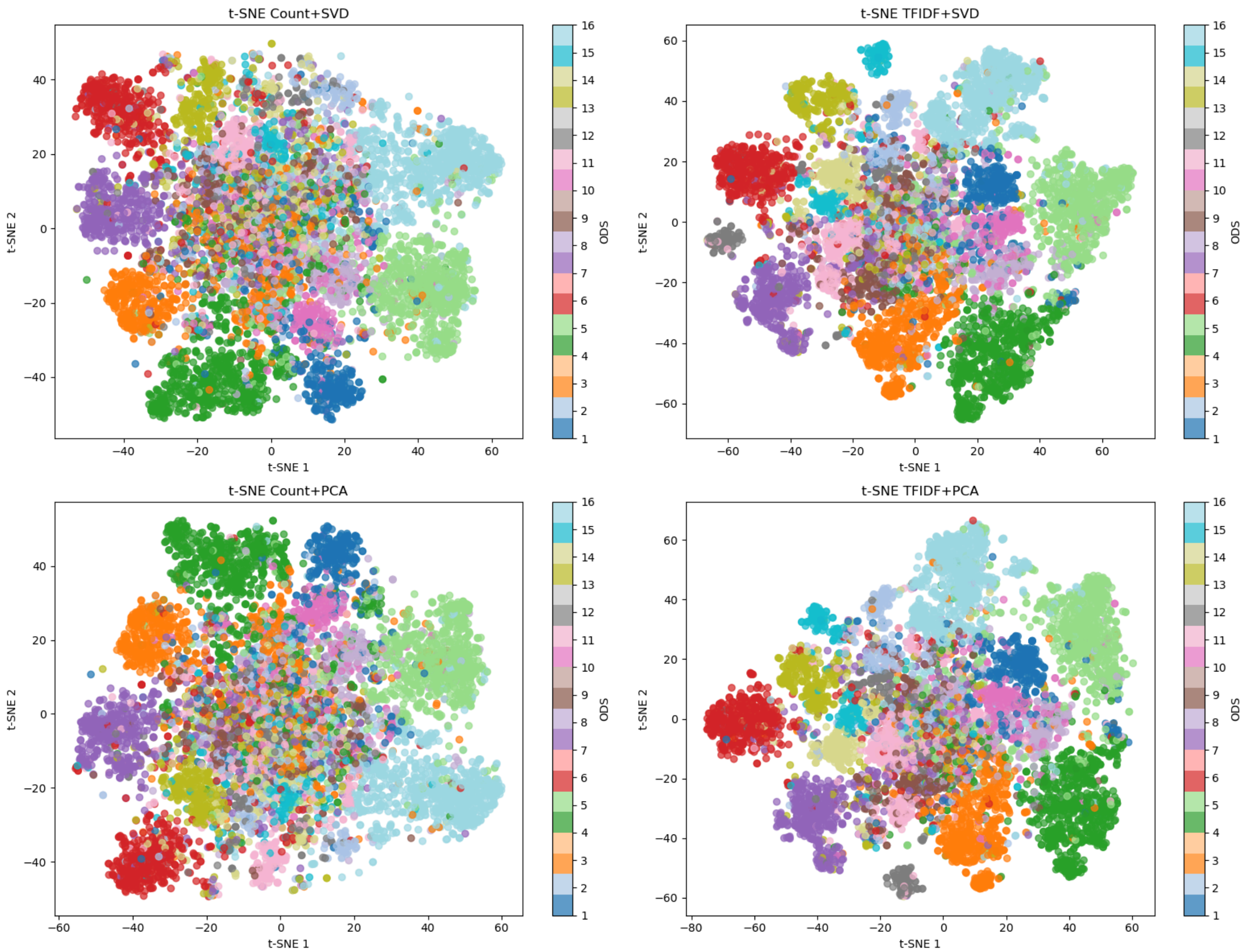
```python
class DataProcessingPipelineFactory:
    def get_pipelines_and_params(self, vectorizer, reduction):

        steps = [
            ('preprocess', TextPreprocessorTransformer())
        ]
        if vectorizer == 'count':
            steps.append(('vectorizer', CountVectorizer()))
        if vectorizer == 'tfidf':
            steps.append(('vectorizer', TfidfVectorizer()))
        if reduction == 'svd':
            steps.append(('dimred', TruncatedSVD(n_components=100)))
        if reduction == 'pca':
            steps.append(('to_dense', ToDenseTransformer()))
            steps.append(('dimred', PCA(n_components=100)))

        pipeline = Pipeline(steps)
        return pipeline
```

In [130…

```python
for bow in bow_list:
    pipeline = DataProcessingPipelineFactory().get_pipelines_and_params(bow['vectorized'], bow['reduction'])
    bow['train_data'] = pipeline.fit_transform(x_train)
    bow['test_data'] = pipeline.transform(x_test)
```

In [131…

```python
nombres = ['Count+SVD', 'TFIDF+SVD', 'Count+PCA', 'TFIDF+PCA']
labels = y_train.astype(int)

tsne_results = []
for bow in bow_list:
    tsne = TSNE(n_components=2, random_state=42)
    tsne_result = tsne.fit_transform(bow['train_data'])
    tsne_results.append(tsne_result)

fig, axes = plt.subplots(2, 2, figsize=(16, 12))
for i, ax in enumerate(axes.flat):
    scatter = ax.scatter(tsne_results[i][:,0], tsne_results[i][:,1], c=labels, cmap='tab20', alpha=0.7)
    ax.set_title(f't-SNE {nombres[i]}')
    ax.set_xlabel('t-SNE 1')
    ax.set_ylabel('t-SNE 2')
    cbar = plt.colorbar(scatter, ax=ax, ticks=range(1,17))
    cbar.set_label('ODS')

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



In [132…

```python
class ModelPipelineFactory:
    def get_pipelines_and_params(self, algorithm):

        if algorithm == 'logistic':
            model = LogisticRegression(max_iter=2000)
            param_grid = {
                'model__C': [0.1, 1, 10],
                'model__solver': ['lbfgs', 'saga']
            }
        elif algorithm == 'randomforest':
            model = RandomForestClassifier()
            param_grid = {
                'model__n_estimators': [100, 200],
                'model__max_depth': [None, 10, 20]
            }
        elif algorithm == 'kneighbors':
            model = KNeighborsClassifier()
            param_grid = {
                'model__n_neighbors': [3, 5, 7],
                'model__weights': ['uniform', 'distance']
            }
        else:
            raise ValueError(f"Algoritmo no soportado: {algorithm}")

        steps = [
            ('model', model)
        ]

        pipeline = Pipeline(steps)
        return pipeline, param_grid
```

In [133…

```python
def print_reports(y_test, y_pred, title):
    # Reporte de clasificación como tabla
    report_dict = classification_report(y_test, y_pred, output_dict=True)
    df_report = pd.DataFrame(report_dict).transpose()

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 6))

    # Tabla de reporte de clasificación
    ax1.axis('off')
    table = ax1.table(cellText=df_report.round(2).values,
                      colLabels=df_report.columns,
                      rowLabels=df_report.index,
```

```python
                     loc='center',
                     cellLoc='center')
    ax1.set_title('Reporte de Clasificación: ' + title)

    # Matriz de confusión con ConfusionMatrixDisplay
    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(ax=ax2, cmap=plt.cm.Blues)
    ax2.set_title('Matriz de Confusión')

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

In [134…

```python
algorithms = ['logistic', 'randomforest', 'kneighbors']
#algorithms = ['kneighbors']
```

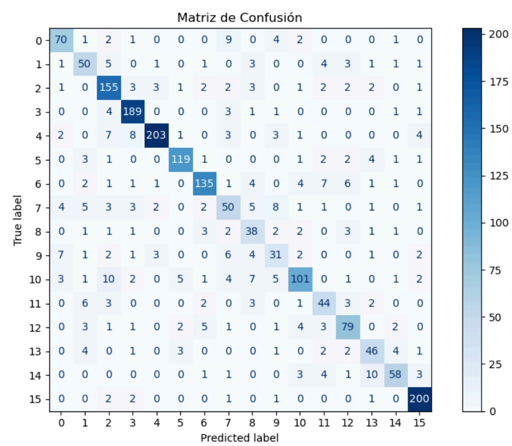
In [135…

```python
for algorithm in algorithms:
    for bow in bow_list:
        title = f"{algorithm} con {bow['vectorized']} + {bow['reduction']}"
        print("ENTRENAMIENTO Y EVALUACION PARA:", title)
        pipelineFactory = ModelPipelineFactory()
        pipeline, params = pipelineFactory.get_pipelines_and_params(algorithm)
        grid = GridSearchCV(pipeline, params, cv=3, scoring='accuracy', n_jobs=-1)
        grid.fit(bow['train_data'], y_train)
        print("Mejores hiperparámetros:", grid.best_params_)
        print("Mejor score de validación:", grid.best_score_)

        y_pred = grid.predict(bow['test_data'])

        print_reports(y_test, y_pred, title)
```

```
ENTRENAMIENTO Y EVALUACION PARA: logistic con count + svd
Mejores hiperparámetros: {'model__C': 0.1, 'model__solver': 'lbfgs'}
Mejor score de validación: 0.8048937973209819
```

Reporte de Clasificación: logistic con count + svd

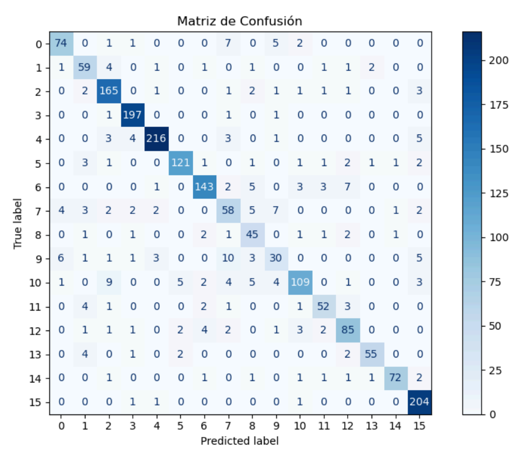
	precision	recall	f1-score	support
1	0.8	0.78	0.79	90.0
2	0.65	0.7	0.68	71.0
3	0.79	0.87	0.83	178.0
4	0.89	0.94	0.92	200.0
5	0.95	0.88	0.91	232.0
6	0.91	0.88	0.89	135.0
7	0.88	0.82	0.85	164.0
8	0.6	0.58	0.59	86.0
9	0.56	0.69	0.62	55.0
10	0.54	0.52	0.53	60.0
11	0.82	0.71	0.76	143.0
12	0.64	0.69	0.66	64.0
13	0.77	0.77	0.77	102.0
14	0.66	0.72	0.69	64.0
15	0.82	0.72	0.76	81.0
16	0.92	0.97	0.94	207.0
accuracy	0.81	0.81	0.81	0.81
macro avg	0.76	0.76	0.76	1932.0
weighted avg	0.82	0.81	0.81	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: logistic con tfidf + svd
Mejores hiperparámetros: {'model__C': 10, 'model__solver': 'lbfgs'}
Mejor score de validación: 0.858234422894617
```

Reporte de Clasificación: logistic con tfidf + svd

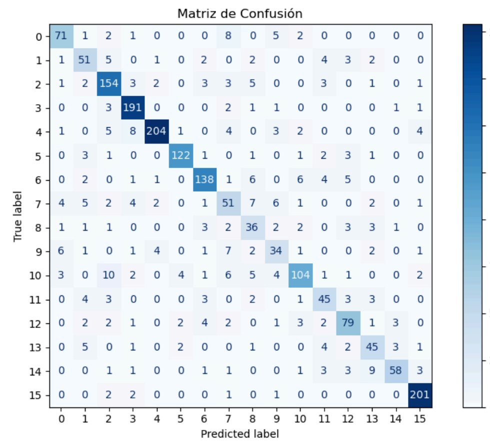
	precision	recall	f1-score	support
1	0.86	0.82	0.84	90.0
2	0.76	0.83	0.79	71.0
3	0.87	0.93	0.9	178.0
4	0.94	0.98	0.96	200.0
5	0.96	0.93	0.95	232.0
6	0.93	0.9	0.91	135.0
7	0.92	0.87	0.89	164.0
8	0.64	0.67	0.66	86.0
9	0.66	0.82	0.73	55.0
10	0.6	0.5	0.55	60.0
11	0.89	0.76	0.82	143.0
12	0.84	0.81	0.83	64.0
13	0.81	0.83	0.82	102.0
14	0.93	0.86	0.89	64.0
15	0.96	0.89	0.92	81.0
16	0.9	0.99	0.94	207.0
accuracy	0.87	0.87	0.87	0.87
macro avg	0.84	0.84	0.84	1932.0
weighted avg	0.87	0.87	0.87	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: logistic con count + pca
/Users/I508111/miniconda3/lib/python3.13/site-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
/Users/I508111/miniconda3/lib/python3.13/site-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
Mejores hiperparámetros: {'model__C': 0.1, 'model__solver': 'saga'}
Mejor score de validación: 0.8016572496184148
```

Reporte de Clasificación: logistic con count + pca

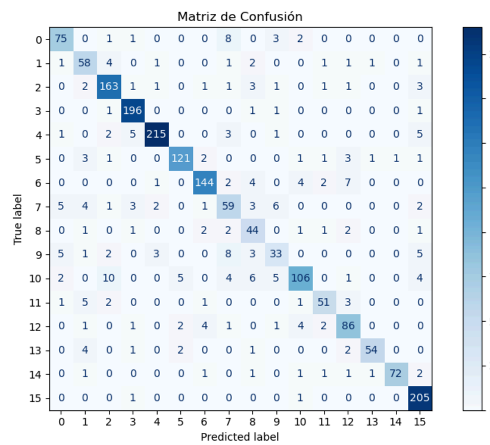
	precision	recall	f1-score	support
1	0.81	0.79	0.8	90.0
2	0.66	0.72	0.69	71.0
3	0.81	0.87	0.83	178.0
4	0.88	0.96	0.92	200.0
5	0.95	0.88	0.91	232.0
6	0.93	0.9	0.92	135.0
7	0.87	0.84	0.86	164.0
8	0.58	0.59	0.59	86.0
9	0.53	0.65	0.59	55.0
10	0.6	0.57	0.58	60.0
11	0.84	0.73	0.78	143.0
12	0.66	0.7	0.68	64.0
13	0.77	0.77	0.77	102.0
14	0.65	0.7	0.68	64.0
15	0.88	0.72	0.79	81.0
16	0.93	0.97	0.95	207.0
accuracy	0.82	0.82	0.82	0.82
macro avg	0.77	0.77	0.77	1932.0
weighted avg	0.82	0.82	0.82	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: logistic con tfidf + pca
Mejores hiperparámetros: {'model__C': 10, 'model__solver': 'saga'}
Mejor score de validación: 0.8600470223771195
```

Reporte de Clasificación: logistic con tfidf + pca

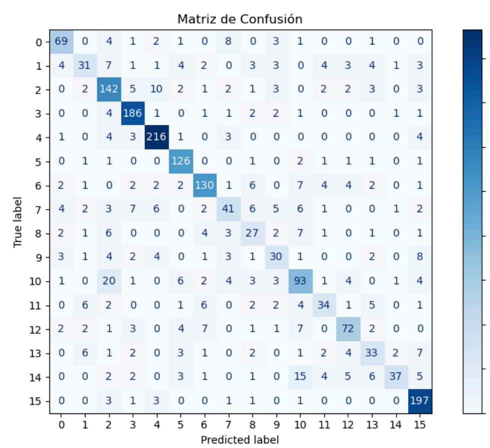
	precision	recall	f1-score	support
1	0.83	0.83	0.83	90.0
2	0.72	0.82	0.77	71.0
3	0.87	0.92	0.89	178.0
4	0.93	0.98	0.96	200.0
5	0.96	0.93	0.95	232.0
6	0.93	0.9	0.91	135.0
7	0.92	0.88	0.9	164.0
8	0.66	0.69	0.67	86.0
9	0.65	0.8	0.72	55.0
10	0.65	0.55	0.59	60.0
11	0.88	0.74	0.8	143.0
12	0.85	0.8	0.82	64.0
13	0.8	0.84	0.82	102.0
14	0.95	0.84	0.89	64.0
15	0.99	0.89	0.94	81.0
16	0.89	0.99	0.94	207.0
accuracy	0.87	0.87	0.87	0.87
macro avg	0.84	0.84	0.84	1932.0
weighted avg	0.87	0.87	0.87	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: randomforest con count + svd
Mejores hiperparámetros: {'model__max_depth': None, 'model__n_estimators': 200}
Mejor score de validación: 0.7428800325887704
```

Reporte de Clasificación: randomforest con count + svd

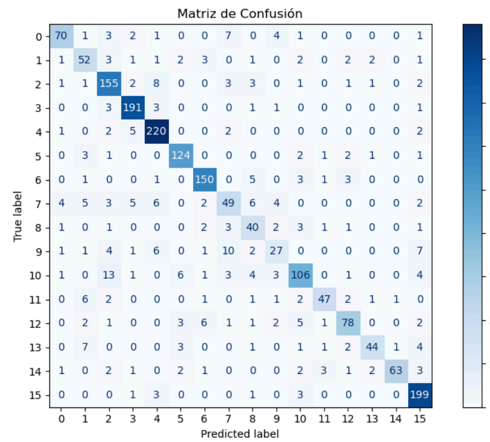
	precision	recall	f1-score	support
1	0.78	0.77	0.78	90.0
2	0.58	0.44	0.5	71.0
3	0.7	0.8	0.74	178.0
4	0.86	0.93	0.89	200.0
5	0.88	0.93	0.91	232.0
6	0.82	0.93	0.88	135.0
7	0.82	0.79	0.81	164.0
8	0.61	0.48	0.54	86.0
9	0.47	0.49	0.48	55.0
10	0.56	0.5	0.53	60.0
11	0.64	0.65	0.64	143.0
12	0.63	0.53	0.58	64.0
13	0.75	0.71	0.73	102.0
14	0.55	0.52	0.53	64.0
15	0.86	0.46	0.6	81.0
16	0.83	0.95	0.89	207.0
accuracy	0.76	0.76	0.76	0.76
macro avg	0.71	0.68	0.69	1932.0
weighted avg	0.75	0.76	0.75	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: randomforest con tfidf + svd
Mejores hiperparámetros: {'model__max_depth': None, 'model__n_estimators': 200}
Mejor score de validación: 0.8241839354460714
```

Reporte de Clasificación: randomforest con tfidf + svd

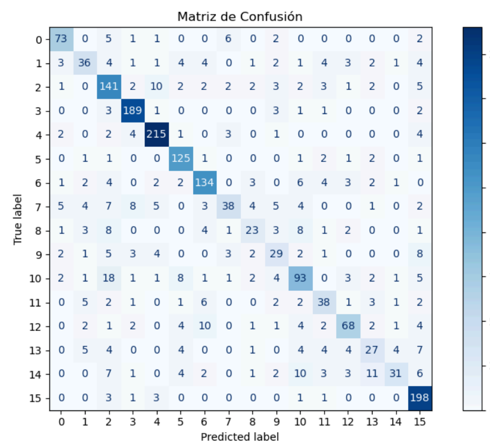
	precision	recall	f1-score	support
1	0.86	0.78	0.82	90.0
2	0.66	0.73	0.69	71.0
3	0.8	0.87	0.84	178.0
4	0.91	0.96	0.93	200.0
5	0.88	0.95	0.91	232.0
6	0.89	0.92	0.9	135.0
7	0.9	0.91	0.91	164.0
8	0.63	0.57	0.6	86.0
9	0.61	0.73	0.66	55.0
10	0.61	0.45	0.52	60.0
11	0.81	0.74	0.77	143.0
12	0.85	0.73	0.79	64.0
13	0.84	0.76	0.8	102.0
14	0.86	0.69	0.77	64.0
15	0.97	0.78	0.86	81.0
16	0.87	0.96	0.91	207.0
accuracy	0.84	0.84	0.84	0.84
macro avg	0.81	0.78	0.79	1932.0
weighted avg	0.84	0.84	0.83	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: randomforest con count + pca
Mejores hiperparámetros: {'model__max_depth': None, 'model__n_estimators': 200}
Mejor score de validación: 0.7339457809360722
```

Reporte de Clasificación: randomforest con count + pca

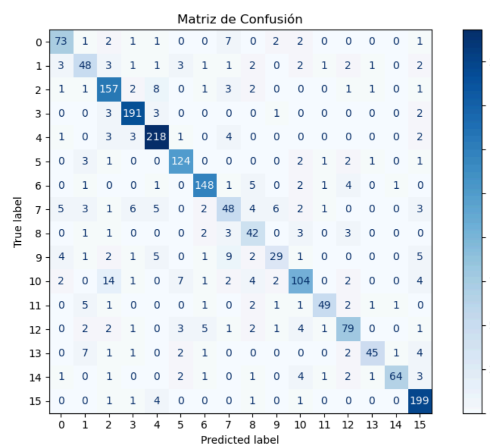
	precision	recall	f1-score	support
1	0.81	0.81	0.81	90.0
2	0.6	0.51	0.55	71.0
3	0.66	0.79	0.72	178.0
4	0.88	0.94	0.91	200.0
5	0.88	0.93	0.91	232.0
6	0.81	0.93	0.86	135.0
7	0.8	0.82	0.81	164.0
8	0.7	0.44	0.54	86.0
9	0.57	0.42	0.48	55.0
10	0.51	0.48	0.5	60.0
11	0.67	0.65	0.66	143.0
12	0.59	0.59	0.59	64.0
13	0.76	0.67	0.71	102.0
14	0.5	0.42	0.46	64.0
15	0.78	0.38	0.51	81.0
16	0.79	0.96	0.86	207.0
accuracy	0.75	0.75	0.75	0.75
macro avg	0.71	0.67	0.68	1932.0
weighted avg	0.75	0.75	0.74	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: randomforest con tfidf + pca
Mejores hiperparámetros: {'model__max_depth': None, 'model__n_estimators': 200}
Mejor score de validación: 0.8258679400426973
```

Reporte de Clasificación: randomforest con tfidf + pca

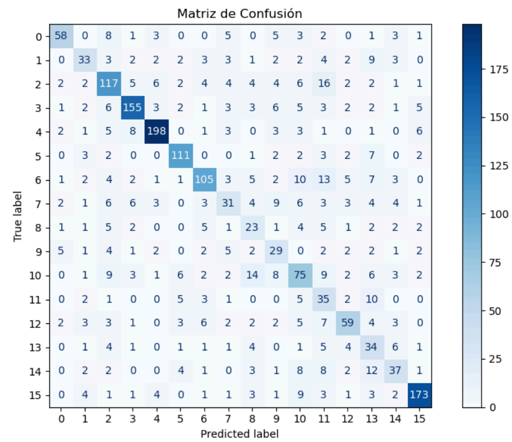
	precision	recall	f1-score	support
1	0.81	0.81	0.81	90.0
2	0.66	0.68	0.67	71.0
3	0.81	0.88	0.85	178.0
4	0.91	0.96	0.93	200.0
5	0.89	0.94	0.91	232.0
6	0.87	0.92	0.9	135.0
7	0.9	0.9	0.9	164.0
8	0.61	0.56	0.58	86.0
9	0.63	0.76	0.69	55.0
10	0.69	0.48	0.57	60.0
11	0.81	0.73	0.77	143.0
12	0.89	0.77	0.82	64.0
13	0.8	0.77	0.79	102.0
14	0.9	0.7	0.79	64.0
15	0.96	0.79	0.86	81.0
16	0.87	0.96	0.91	207.0
accuracy	0.84	0.84	0.84	0.84
macro avg	0.81	0.79	0.8	1932.0
weighted avg	0.84	0.84	0.83	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: kneighbors con count + svd
Mejores hiperparámetros: {'model__n_neighbors': 7, 'model__weights': 'distance'}
Mejor score de validación: 0.6418952784001327
```

Reporte de Clasificación: kneighbors con count + svd

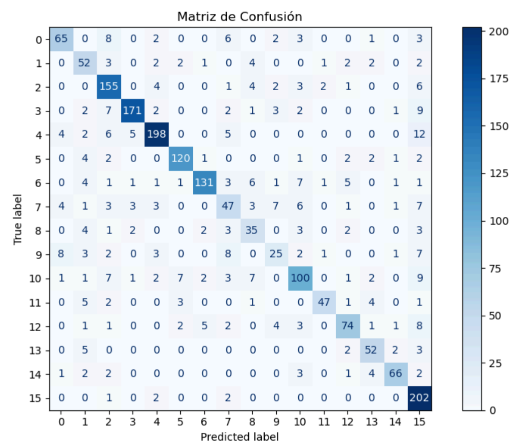
	precision	recall	f1-score	support
1	0.78	0.64	0.71	90.0
2	0.56	0.46	0.51	71.0
3	0.65	0.66	0.65	178.0
4	0.82	0.78	0.8	200.0
5	0.89	0.85	0.87	232.0
6	0.81	0.82	0.82	135.0
7	0.76	0.64	0.7	164.0
8	0.48	0.36	0.41	86.0
9	0.33	0.42	0.37	55.0
10	0.39	0.48	0.43	60.0
11	0.52	0.52	0.52	143.0
12	0.29	0.55	0.38	64.0
13	0.66	0.58	0.62	102.0
14	0.32	0.53	0.4	64.0
15	0.54	0.46	0.49	81.0
16	0.88	0.84	0.86	207.0
accuracy	0.66	0.66	0.66	0.66
macro avg	0.61	0.6	0.6	1932.0
weighted avg	0.68	0.66	0.67	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: kneighbors con tfidf + svd
Mejores hiperparámetros: {'model__n_neighbors': 7, 'model__weights': 'distance'}
Mejor score de validación: 0.7752461634015032
```

Reporte de Clasificación: kneighbors con tfidf + svd

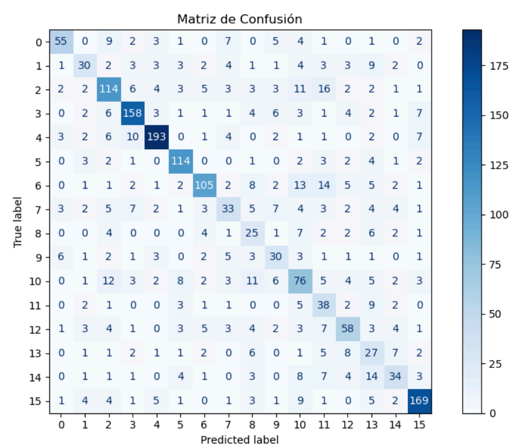
	precision	recall	f1-score	support
1	0.78	0.72	0.75	90.0
2	0.6	0.73	0.66	71.0
3	0.77	0.87	0.82	178.0
4	0.93	0.86	0.89	200.0
5	0.9	0.85	0.88	232.0
6	0.89	0.89	0.89	135.0
7	0.92	0.8	0.86	164.0
8	0.57	0.55	0.56	86.0
9	0.57	0.64	0.6	55.0
10	0.57	0.42	0.48	60.0
11	0.75	0.7	0.72	143.0
12	0.9	0.73	0.81	64.0
13	0.8	0.73	0.76	102.0
14	0.76	0.81	0.79	64.0
15	0.89	0.81	0.85	81.0
16	0.73	0.98	0.83	207.0
accuracy	0.8	0.8	0.8	0.8
macro avg	0.77	0.76	0.76	1932.0
weighted avg	0.8	0.8	0.8	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: kneighbors con count + pca
Mejores hiperparámetros: {'model__n_neighbors': 7, 'model__weights': 'distance'}
Mejor score de validación: 0.6378824339989388
```

Reporte de Clasificación: kneighbors con count + pca

	precision	recall	f1-score	support
1	0.76	0.61	0.68	90.0
2	0.55	0.42	0.48	71.0
3	0.66	0.64	0.65	178.0
4	0.8	0.79	0.79	200.0
5	0.88	0.83	0.85	232.0
6	0.79	0.84	0.81	135.0
7	0.78	0.64	0.7	164.0
8	0.49	0.38	0.43	86.0
9	0.32	0.45	0.38	55.0
10	0.45	0.5	0.48	60.0
11	0.49	0.53	0.51	143.0
12	0.35	0.59	0.44	64.0
13	0.6	0.57	0.58	102.0
14	0.27	0.42	0.33	64.0
15	0.53	0.42	0.47	81.0
16	0.84	0.82	0.83	207.0
accuracy	0.65	0.65	0.65	0.65
macro avg	0.6	0.59	0.59	1932.0
weighted avg	0.67	0.65	0.66	1932.0




```
ENTRENAMIENTO Y EVALUACION PARA: kneighbors con tfidf + pca
Mejores hiperparámetros: {'model__n_neighbors': 7, 'model__weights': 'distance'}
Mejor score de validación: 0.7687732188703063
```

Reporte de Clasificación: kneighbors con tfidf + pca

	precision	recall	f1-score	support
1	0.79	0.7	0.74	90.0
2	0.64	0.73	0.68	71.0
3	0.74	0.87	0.8	178.0
4	0.92	0.86	0.89	200.0
5	0.9	0.84	0.87	232.0
6	0.86	0.89	0.88	135.0
7	0.88	0.79	0.83	164.0
8	0.56	0.49	0.52	86.0
9	0.54	0.6	0.57	55.0
10	0.52	0.35	0.42	60.0
11	0.76	0.65	0.7	143.0
12	0.91	0.77	0.83	64.0
13	0.81	0.75	0.78	102.0
14	0.75	0.75	0.75	64.0
15	0.93	0.78	0.85	81.0
16	0.69	0.98	0.81	207.0
accuracy	0.78	0.78	0.78	0.78
macro avg	0.76	0.74	0.74	1932.0
weighted avg	0.79	0.78	0.78	1932.0

Matriz de Confusión

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	63	1	5	1	3	0	0	7	1	2	2	0	0	0	0	5
1	0	52	3	0	1	2	1	0	3	1	0	1	2	2	0	3
2	1	0	155	2	4	0	1	1	3	1	0	2	0	1	0	7
3	0	0	9	172	2	0	0	2	1	2	0	0	0	1	0	11
4	3	1	7	4	194	0	0	6	0	2	2	0	0	0	0	13
5	0	4	2	0	0	120	0	1	1	0	0	0	1	2	1	3
6	0	3	3	1	1	1	130	3	5	0	5	1	6	0	0	5
7	3	2	6	2	4	2	3	42	3	8	3	0	1	1	0	6
8	0	2	2	2	0	0	4	1	33	0	6	0	2	0	0	3
9	8	2	2	1	3	0	0	8	0	21	3	0	0	1	0	11
10	2	3	11	0	2	7	2	1	6	1	93	1	1	0	0	13
11	0	5	1	0	0	3	0	0	2	0	0	49	2	2	0	0
12	0	1	1	0	0	2	5	1	1	1	6	0	76	1	1	6
13	0	4	0	1	0	1	1	1	1	0	0	0	2	48	3	2
14	0	1	1	0	0	1	1	0	1	1	3	0	1	5	63	3
15	0	0	1	0	2	0	0	1	0	0	0	0	0	0	0	203

In [ ]: