DATOS UTILIZADOS

EJERCICIO 10 - Clasificación Multiclase Iris Dataset

Archivo fuente: `EjerciciosPropuestos/Ejercicio10/Ejercicio10_Clasificación_Multiclase_Iris.ipynb` **Dataset:** Iris dataset (sklearn.datasets.load_iris)

Resultados de Clasificación Obtenidos:

python

```
--- Logistic Regression ---
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45


--- SVM ---
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45


--- KNN ---
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
```

```
    macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Configuración utilizada:

- Test size: 30% (45 muestras de prueba)
- Random state: 42
- Modelos: LogisticRegression(max_iter=200), SVC(), KNeighborsClassifier()

EJERCICIO 11 - GridSearchCV Random Forest

Archivo fuente: `EjerciciosPropuestos/Ejercicio11/Ejercicio11_Validación_Cruzada_y_Ajuste_de_Hiperparámetros_Random _Forest.ipynb` **Dataset:** `titanic_limpio.csv`

Parámetros y Resultados Obtenidos:

python

```
# Parámetros de búsqueda:
param_grid = {'n_estimators': [50, 100], 'max_depth': [4, 6, 8]}


# Mejores parámetros encontrados:
Mejores parámetros: {'max_depth': 4, 'n_estimators': 100}


# Reporte de clasificación final:
              precision    recall  f1-score   support

           0       0.87      0.99      0.92       967
           1       0.94      0.57      0.71       342

    accuracy                           0.88      1309
   macro avg       0.90      0.78      0.81      1309
weighted avg       0.88      0.88      0.87      1309
```

Configuración utilizada:

- Validación cruzada: cv=5
- Dataset completo: 1309 registros
- Variables predictoras: Age, Fare, sibsp, Parch, Pclass, Embarked, Sex_male
- Variable objetivo: 2urvived (columna renombrada de Survived)

EJERCICIO 12 - Detección de Spam TF-IDF + SVM

Archivos fuente:

- `EjerciciosPropuestos/Ejercicio12/Ejercicio12_Detección_de_Spam_TF_IDF_SVM.ipynb`
- `EjerciciosPropuestos/Ejercicio12/Ejercicio12_Detección_de_Spam_TF_IDF_SVM_data_rt.ipynb`

Datasets:

1. `spam.csv` - Dataset específico de detección de spam
2. `data_rt.csv` - Dataset de reseñas de Rotten Tomatoes

Resultados Obtenidos:

Ejecución 1 - Dataset spam.csv:

python

```
# Configuración:
df = pd.read_csv('../../Dataset/spam.csv')
vectorizer = TfidfVectorizer(stop_words='english', max_features=3000)

# Resultados:
                precision    recall  f1-score   support

           ham       0.98      1.00      0.99      1453
          spam       0.99      0.84      0.91       219

    accuracy                           0.98      1672
   macro avg       0.99      0.92      0.95      1672
weighted avg       0.98      0.98      0.98      1672
```

Ejecución 2 - Dataset data_rt.csv:

python

```
# Configuración:
df = pd.read_csv('../../Dataset/data_rt.csv')
vectorizer = TfidfVectorizer(stop_words='english', max_features=3000)

# Resultados:
              precision    recall  f1-score   support

           0       0.74      0.74      0.74      1615
           1       0.74      0.74      0.74      1584

    accuracy                           0.74      3199
   macro avg       0.74      0.74      0.74      3199
```

weighted avg 0.74 0.74 0.74 3199

Análisis comparativo:

- **Spam detection:** 98% accuracy - SVM altamente efectivo
- **Sentiment analysis:** 74% accuracy - SVM moderadamente efectivo
- **Diferencia clave:** Patrones estructurados de spam vs complejidad de sentimientos

Configuración común:

- División: test_size=0.3, random_state=42
- Modelo: SVC() con parámetros default
- Vectorización: TF-IDF con 3000 características máximas

EJERCICIO 13 - PCA Netflix

Archivo fuente: Según documento teórico del ejercicio **Dataset:** `netflix_titles.csv`

Configuración Teórica del Ejercicio:

python

```
# Procesamiento de datos:
df['duration_num'] = df['duration'].str.replace(' min', '').astype(float)
X = df[['duration_num']].fillna(0)

# PCA aplicado:
pca = PCA(n_components=1)
X_pca = pca.fit_transform(X)

# Visualización:
plt.hist(X_pca, bins=30)
plt.title('Distribución de componentes principales')
```

Datos esperados:

- Variable numérica: duración en minutos de películas
- Componente principal: dirección de mayor varianza en duraciones
- Histograma: distribución de películas según componente principal

EJERCICIO 14 - KMeans Clustering Opiniones

Archivo fuente: `EjerciciosPropuestos/Ejercicio14/Ejercicio14_Clustering KMeans Opiniones.ipynb` **Dataset:** `data_rt.csv` (Rotten Tomatoes reviews)

Datos y Resultados Obtenidos:

python

```
# Configuración:
vectorizer = TfidfVectorizer(stop_words='english')
kmeans = KMeans(n_clusters=2, random_state=42)

# Resultado del DataFrame con clusters:
                                                reviews  labels  cluster
0                simplistic , silly and tedious .       0       0
1  it's so laddish and juvenile , only teenage bo...       0       0
2  exploitative and largely devoid of the depth o...       0       0
3  [garbus] discards the potential for pathologic...       0       0
4  a visually flashy but narratively opaque and e...       0       0
```

Observación clave:

- Las primeras 5 muestras mostradas tienen labels=0 (negativo) y cluster=0
- Esto sugiere que el clustering está capturando el patrón de sentimiento negativo
- Dataset completo: miles de reviews de películas

EJERCICIO 15 - Árbol de Decisión Visualización

Archivo fuente: `EjerciciosPropuestos/Ejercicio15/Ejercicio15_Árbol de Decisión Graphviz.ipynb` **Dataset:** `titanic_limpio.csv`

Código y Configuración Utilizados:

python

```
# Configuración del modelo:
model = DecisionTreeClassifier(max_depth=4)
model.fit(X, y)

# Exportación visual:
dot_data = export_graphviz(model, out_file=None,
                           feature_names=X.columns,
                           class_names=['No', 'Sí'],
                           filled=True)
graph = graphviz.Source(dot_data)

# Resultado:
```

```
graph.render("titanic_tree", format="png", cleanup=True)
# Output: 'titanic_tree.png'
```

Datos del modelo:

- Variables predictoras: Age, Fare, sibsp, Parch, Pclass, Embarked, Sex_male
- Variable objetivo: 2urvived
- Profundidad máxima: 4 niveles
- Clases: ['No', 'Sí'] (No sobrevivió, Sí sobrevivió)

DATASETS IDENTIFICADOS EN EL PROYECTO

Datasets Disponibles:

1. **`titanic_limpio.csv`** - Utilizado en ejercicios 11 y 15
   - Ubicación: `EjerciciosPropuestos/Ejercicio1/titanic_limpio.csv`
   - Columnas: Age, Fare, sibsp, Parch, Pclass, Embarked, 2urvived, Sex_male
2. **`netflix_titles.csv`** - Para ejercicio 13
   - Ubicación: `Dataset/netflix_titles.csv`
   - Columnas incluyen: duration, type, title, etc.
3. **`data_rt.csv`** - Utilizado en ejercicio 14
   - Ubicación: `Dataset/data_rt.csv`
   - Columnas: reviews, labels
4. **Iris Dataset** - Utilizado en ejercicio 10
   - Fuente: sklearn.datasets.load_iris()
   - Built-in dataset, no archivo físico requerido

Archivos Generados:

- `titanic_tree.png` - Visualización del árbol de decisión
- Matrices TF-IDF en memoria para clustering y clasificación
- Componentes principales PCA para análisis de Netflix

Scripts de Soporte Identificados:

- `generar_csv.py` - Para crear datasets sintéticos
- `crear_spam_csv_desde_dataset_original.py` - Para generar spam.csv
- `setup_ejercicios.py` - Para configurar estructura de directorios