UNIVERSIDAD NACIONAL DE CAÑETE

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

GUÍA DE PRACTICA:

PROCESAMIENTO DE DATOS

DOCENTE:

MAGALY ROXANA ARANGÜENA YLLANES

ASIGNATURA:

MACHINE LEARNING

JEFE DE PRÁCTICA:

(NOMBRES Y APELLIDOS)

San Vicente, Cañete 2025

UNIVERSIDAD NACIONAL DE CAÑETE

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

UNIVERSIDAD NACIONAL DE CAÑETE

FACULTAD DE INGENIERÍA DE SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

GUIA DE PRACTICA N°. 01					
Fecha:				**Semana:**	2
Curso:	Machine Learning	**Ciclo:**	VII	**Turno y sección:**	
Contenido temático:	PROCESAMIENTO DE DATOS				
Laboratorio:	SI			**Duración:**	4 horas
Docente:	M.Sc. Magaly Roxana Arangüena Yllanes				
Jefes de practica:					
Alumno(a) o Grupo:				**Nota**	

OBJETIVO

Objetivo de la Práctica

Aplicar técnicas fundamentales de limpieza y transformación de datos utilizando el dataset Titanic, con el propósito de preparar adecuadamente la información para el entrenamiento de modelos de Machine Learning.

ACTIVIDADES

PROCESAMIENTO DE DATOS

1. LIMPIEZA DE DATOS

Eliminación, reemplazo de valores nulos e imputación de datos

Paso 1: importar librerías

```
import pandas as pd
import numpy as np
```

Paso 2: eliminar columnas

```
# Eliminar columnas irrelevantes: PassengerId, Name, Ticket (no ayudan a predecir)
df = df.drop(columns=['PassengerId', 'Name', 'Ticket'])


# Eliminar Cabin por exceso de nulos (>70%)
df = df.drop(columns=['Cabin'])

print(df.head())
```

Resultado

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

Paso 3: reemplazar los valores nulos por NAN

```
#reemplazar los valores nulos por NAN
df.replace('', np.nan, inplace=True)
print(df.head())
print(df.isnull().sum())
```

Resultado

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

Paso 4: Convertir Sexo, embarque y tipo en categoría explicito

```
# Convertir Sex y Embarked a tipo categórico explícito
df['Sex'] = df['Sex'].astype('category')
df['Embarked'] = df['Embarked'].astype('category')
df['Pclass'] = df['Pclass'].astype('category')

print(df.head())
```

Resultado

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

Nota los resultados no se encuentran visibles porque se muestra solamente los 5 primeros registros

Paso 5: Imputación de edad y embarque

```
# Imputar Age con la mediana (variable continua con outliers)
df['Age'] = df['Age'].fillna(df['Age'].median())

# Imputar Embarked con la moda (categórica)
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
print(df.head())
```

Resultado

```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

2. TRANSFORMACION DE DATOS

normalizar variables continuas, codificar variables categóricas.

Paso 6: Importar librerías

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
```

Paso 7: separar características

```
# Separar características
X = df.drop('Survived', axis=1)
y = df['Survived']
```

Paso 8: estandarizar por tipos de variables

```
# Variables numéricas a estandarizar (mejor que normalizar por outliers)
numerical_features = ['Age', 'Fare']
numerical_transformer = StandardScaler()

# Variables categóricas a codificar con One-Hot
categorical_features = ['Sex', 'Embarked', 'Pclass']
categorical_transformer = OneHotEncoder(handle_unknown='ignore')
```

Nota: No se estandarizaron `SibSp` y `Parch` porque son variables enteras pequeñas, discretas y no continuas, y en muchos casos no afectan el rendimiento si se dejan tal como están. Sin embargo, sí se puede incluirlas si el modelo lo requiere. Se podría estandarizar si se utiliza el modelo (KNN, SVM o PCA-reducción de dimensiones)

Paso 9: Transformar por columna

```
# Transformador combinado
preprocessor = ColumnTransformer(transformers=[
    ('num', numerical_transformer, numerical_features),
    ('cat', categorical_transformer, categorical_features)
])

# Aplicar transformaciones
X_processed = preprocessor.fit_transform(X)
```

Resultado

```
        Age      Fare  Sex_female  Sex_male  Embarked_C  Embarked_Q  \
0 -0.565736 -0.502445         0.0       1.0         0.0         0.0
1  0.663861  0.786845         1.0       0.0         1.0         0.0
2 -0.258337 -0.488854         1.0       0.0         0.0         0.0
3  0.433312  0.420730         1.0       0.0         0.0         0.0
4  0.433312 -0.486337         0.0       1.0         0.0         0.0

   Embarked_S  Pclass_1  Pclass_2  Pclass_3
0         1.0       0.0       0.0       1.0
1         0.0       1.0       0.0       0.0
2         1.0       0.0       0.0       1.0
3         1.0       1.0       0.0       0.0
4         1.0       0.0       0.0       1.0
```

3. SELECCIÓN DE CARACTERÍSTICAS

Identificar las variables más relevantes para el modelo.
Paso 10: Selección de características
En este paso no aplicaremos porque todas las características o columnas son importantes

4. INGENIERIA DE CARACTERISTICAS

Crear nuevas variables que puedan mejorar el rendimiento del modelo.

Paso 11: crear variables unificadas

```
# Crear variable "FamilySize"
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1  # El +1 incluye al propio pasajero

# También podríamos crear: Alone = 1 si no tiene familiares a bordo
df['Alone'] = (df['FamilySize'] == 1).astype(int)
```

5. MUESTREO DE CLASES

Balancear las clases para evitar sesgos en el modelo.
Paso 12: Muestreo de clases

```python
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

# Reaplicar transformador incluyendo nuevas columnas
X = df.drop('Survived', axis=1)
y = df['Survived']

X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Procesar solo entrenamiento antes de aplicar SMOTE
X_train_processed = preprocessor.fit_transform(X_train_raw)

# Aplicar SMOTE solo al conjunto de entrenamiento
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train_processed, y_train)

#antes
print("Distribución original en y_train:")
print(y_train.value_counts())
#despues
print("Distribución después de SMOTE:")
print(pd.Series(y_resampled).value_counts())
```

Resultado

```
Distribución original en y_train:
Survived
0    392
1    231
Name: count, dtype: int64
Distribución después de SMOTE:
Survived
1    392
0    392
Name: count, dtype: int64
```

Paso 13: Unificar columnas

```
# Aplicar preprocesamiento al conjunto completo
X_full = df.drop('Survived', axis=1)
y_full = df['Survived']
X_processed = preprocessor.fit_transform(X_full)

# Obtener nombres de columnas generadas por OneHotEncoder
encoded_columns = preprocessor.named_transformers_['cat'].get_feature_names_out(categorical_features)

# Unir con columnas numéricas
import numpy as np
processed_columns = list(encoded_columns) + numerical_features

# Crear DataFrame resultante
df_encoded = pd.DataFrame(X_processed.toarray() if hasattr(X_processed, 'toarray') else X_processed, columns=processed_columns)

# Agregar variables que no pasaron por ColumnTransformer
df_encoded['FamilySize'] = df['FamilySize'].values
df_encoded['Alone'] = df['Alone'].values

# Agregar la variable objetivo
df_encoded['Survived'] = y_full.values

# Redondear a 1 decimal
df_encoded = df_encoded.round(1)

# Mostrar muestra del nuevo dataset
print(df_encoded.head())
```

Resultado

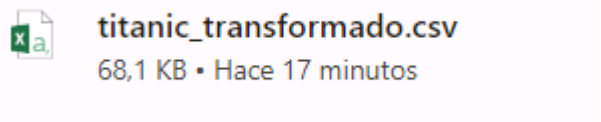
	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S	Pclass_1	\
0	-0.6	-0.5	0.0	1.0	0.0	0.0	
1	0.7	0.8	1.0	0.0	1.0	0.0	
2	-0.3	-0.5	1.0	0.0	0.0	0.0	
3	0.4	0.4	1.0	0.0	0.0	0.0	
4	0.4	-0.5	0.0	1.0	0.0	0.0	

	Pclass_2	Pclass_3	Age	Fare	FamilySize	Alone	Survived
0	1.0	0.0	0.0	1.0	2	0	0
1	0.0	1.0	0.0	0.0	2	0	1
2	1.0	0.0	0.0	1.0	1	1	1
3	1.0	1.0	0.0	0.0	2	0	1
4	1.0	0.0	0.0	1.0	1	1	0

Paso 14: Descargar la data set en .csv

```
# Exportar a CSV
df_encoded.to_csv("titanic_transformado.csv", index=False)

# Si estás en Google Colab y quieres descargarlo al computador:
from google.colab import files
files.download("titanic_transformado.csv")
```

Resultado



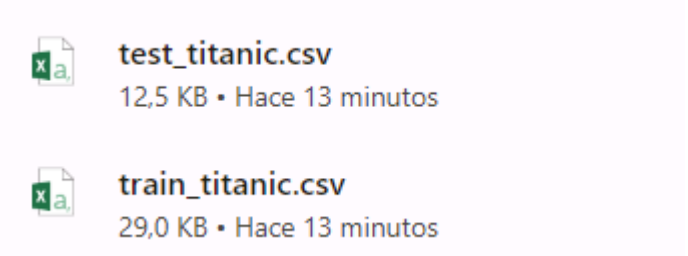
6. PARTICIÓN DE DATOS

Dividir en datos de entrenamiento y datos de prueba

Paso 15:

```
from sklearn.model_selection import train_test_split

# Separar características y etiqueta
X = df_encoded.drop('Survived', axis=1)
y = df_encoded['Survived']

# Dividir en 70% entrenamiento y 30% prueba
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# Reconstruir el DataFrame de entrenamiento (X + y)
df_train = pd.concat([X_train, y_train], axis=1)

# En el conjunto de prueba solo dejamos X (sin Survived)
df_test = X_test.copy()  # sin necesidad de añadir y_test

# Guardar como archivos CSV
df_train.to_csv("train_titanic.csv", index=False)
df_test.to_csv("test_titanic.csv", index=False)

# Descargar si estás en Google Colab
from google.colab import files
files.download("train_titanic.csv")
files.download("test_titanic.csv")
```

Resultado



7. CONCLUSIONES

- La limpieza y transformación adecuada de los datos es fundamental para mejorar la calidad del modelo. Eliminar columnas irrelevantes, imputar valores nulos y convertir variables categóricas en numéricas permitió preparar el dataset para algoritmos de aprendizaje automático.

- La ingeniería de características añadió valor predictivo al modelo, al crear variables como FamilySize y Alone, que capturan aspectos sociales del viaje que no estaban directamente representados en las variables originales.

- El balanceo de clases mediante SMOTE permitió abordar el sesgo hacia la clase mayoritaria, lo cual es especialmente importante en contextos donde se desea detectar correctamente a los sobrevivientes, sin sacrificar la sensibilidad del modelo.

8. RECOMENDACIONES

- Utilizar técnicas de visualización y análisis exploratorio detallado antes del modelado, como ydata-profiling, para identificar valores atípicos, nulos y relaciones relevantes entre variables.

- Comparar modelos entrenados con y sin técnicas de balanceo (como SMOTE), utilizando métricas adecuadas como f1-score y AUC, para tomar decisiones informadas según el objetivo del problema (precisión general vs. detección de clases minoritarias).

- Documentar y versionar cada etapa del preprocesamiento, incluyendo transformaciones y selección de variables, para asegurar reproducibilidad y facilitar mejoras iterativas en proyectos reales de Machine Learning.

ACTIVIDAD GRUPAL

A partir del análisis y exploración de datos realizado en la clase anterior de manera grupal, continúen con el desarrollo del proyecto aplicando esta vez el procesamiento de datos, siguiendo los seis pasos fundamentales de esta etapa.

Nota:

- Solo un integrante del grupo debe realizar el envío
- Adjuntar informe en PDF (máximo 15 páginas)
- Archivos generados (train y test en formato .csv)