

# Proceso de Limpieza y Preparación de Datos de Vinos

## 1. Selección del Dataset Inicial

El conjunto de datos original incluía las siguientes variables clave:

### Categorías geográficas:

- País (country)
- Provincia (province)
- Región 1 (region\_1)
- Región 2 (region\_2)
- Variedad (variety)
- Bodega (winery)
- Denominación (designation)
- Puntuación (points)
- Precio (price)
- Nombre del evaluador (taster\_name)
- Twitter del evaluador (taster\_twitter\_handle)
- Análisis detallado (description)

## ✍ 2. Proceso Integral de Limpieza

### a. Estandarización de Datos

- Renombrado sistemático de columnas para garantizar consistencia
- Normalización de formatos en campos textuales

```
def standardize_column_names(df):  
    """  
        Estandariza los nombres de columnas a minúsculas, sin espacios ni  
        símbolos.  
    """  
    df.columns = (  
        df.columns  
        .str.strip()  
        .str.lower()  
        .str.replace(" ", "_")  
        .str.replace(r"^[^\w_]", "", regex=True)  
    )  
  
    print("📋 [COLUMNS] Nombres de columnas estandarizados.")  
    print("📋 [INFO] Columnas actuales en el dataset:\n",  
df.columns.tolist())  
  
    return df
```

### b. Eliminación de Duplicados

- Identificación y eliminación de entradas repetidas
- Implementación de algoritmos de detección de similitudes para casos complejos

```

def eliminar_duplicados(df):
    """
    Elimina filas duplicadas y reinicia el índice.
    """
    original_shape = df.shape[0]
    df.drop_duplicates(subset=['title'], inplace=True)
    print(f"✅ Duplicados eliminados. Filas eliminadas: {original_shape - df.shape[0]}")
    return df

```

### c. Depuración de Columnas No Esenciales

Eliminación de las siguientes columnas por alto porcentaje de valores nulos o relevancia limitada:

- region\_1 y region\_2 (duplicidad geográfica)
- designation (denominación específica)
- taster\_twitter\_handle (información redundante)

```

def drop_irrelevant_columns(df):
    """
    Elimina columnas vacías, constantes y seleccionadas manualmente.
    """
    protected_cols = []

```

```

    manual_drop = ["region_1", "region_2", "taster_twitter_handle",
"designation"]

    empty_cols = df.columns[df.isnull().all()].tolist()
    constant_cols = df.columns[df.nunique() <= 1].tolist()

    all_to_drop = list(set(empty_cols + constant_cols + manual_drop))
    final_to_drop = [col for col in all_to_drop if col in df.columns and col
not in protected_cols]

    print("☒ Columnas a eliminar:", final_to_drop)

    df = df.drop(columns=final_to_drop, errors='ignore')
    print("☑ Limpieza de columnas irrelevantes completada.")

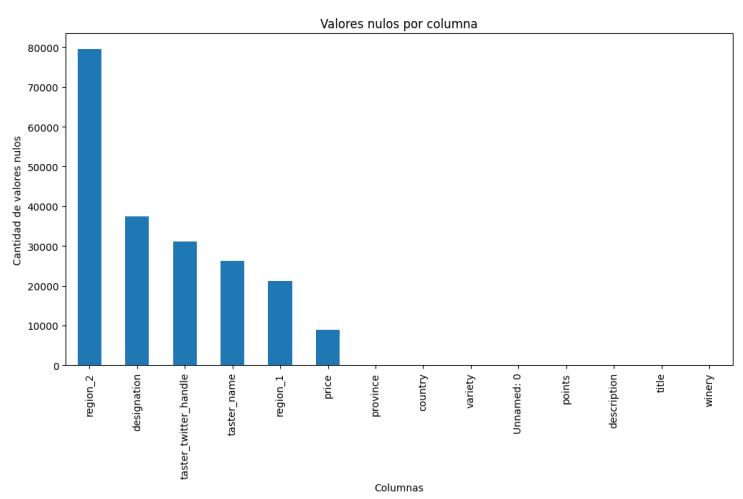
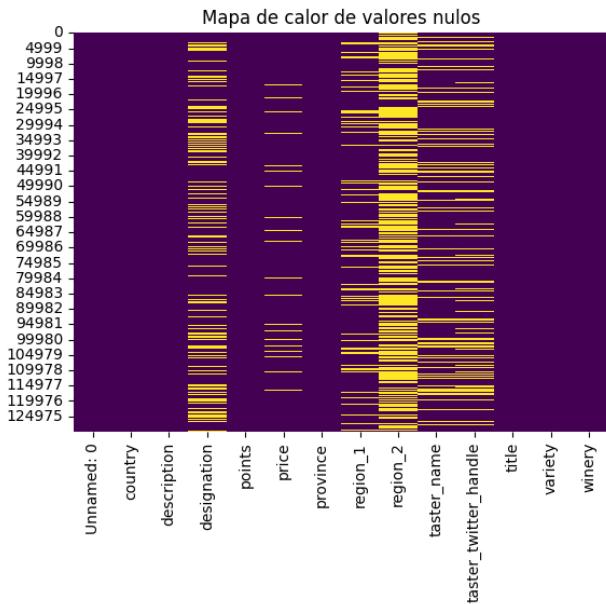
    return df

```

## 3. Gestión de Valores Nulos

### Identificación de valores nulos

- Patrón de valores nulos para detectar correlaciones y patrones Mnar Mar Mcar



En esta imagen podemos observar que tenemos varios registros con bastantes nulos, debido a una no necesaria relevancia en el análisis y a una difícil futura gestión de estos nulos, hemos eliminado las columnas previamente mencionadas en el apartado de depuración de columnas no esenciales.

La única columna con registros vacíos es price, por lo que viendo el porcentaje que suponen respecto al total (7%) optaremos por una técnica sencilla de imputación.

### Tratamiento del Precio (price)

- **Estrategia principal:** Imputación mediante mediana agrupada por combinación de variedad (variety) y provincia (province), debido a bajo porcentaje de nulos en la columna
- **Caso excepcional:** Para combinaciones sin datos suficientes, aplicación de la mediana general del dataset

```
# Imputar precio por grupo
df['price'] = df['price'].fillna(
    df.groupby(['variety', 'province'])['price'].transform('median')
)
df['price'] = df['price'].fillna(df['price'].median())
```

### Tratamiento de nombre de catador (taster\_name)

- **Estrategia principal:** Imputaremos aplicando unknown ya que es una variable categorica la cual no tenemos forma de predecir o suplantar sin obviar la integridad del dataset por completo, asi que este metodo sera como aplicar un flag donde simplemente si se tiene que trabajar con esa columna sabremos que registros no usar

## ⌚ Extracción y Procesamiento de Sabores mediante NLP

### ⚙️ Metodología Implementada

Procesamiento avanzado del texto descriptivo utilizando spaCy:

- Tokenización especializada para terminología enológica
- Filtrado por categorías gramaticales relevantes:
  - Adjetivos (ADJ) descriptivos
  - Sustantivos (NOUN) con longitud mínima de 3 caracteres

```
def extraer_sabores(texto):
    doc = nlp(texto.lower())
    return [token.lemma_ for token in doc
            if token.pos_ in ["ADJ", "NOUN"] and len(token.lemma_) > 3]
```

Aplicamos esta función a los valores de la columna

### 🎯 Objetivo Analítico

- Identificación sistemática de términos relacionados con percepciones sensoriales
- Transformación de descripciones textuales en variables estructuradas (dummies)
- Creación de base sólida para análisis descriptivos y modelos predictivos



## Resultados: 50 Descriptores Sensoriales Más Relevantes

#	Descriptor	Frecuencia
0	wine	82,911
1	flavor	68,737
2	fruit	63,373
3	palate	37,971
4	aroma	36,999
5	acidity	34,890
6	tannin	32,935
7	cherry	32,810
8	finish	32,594
9	black	28,980
...	...	...
46	chocolate	8,508
47	currant	8,492
48	character	8,448
49	vineyard	7,875



## Depuración de Términos No Sensoriales

**Eliminación manual de términos no representativos de sabores u olores:**

- Términos genéricos: *wine, palate, note, nose*
- Referencias temporales: *year, vintage*
- Términos técnicos: *blend, vineyard, character*
- Calificativos subjetivos: *good, great, excellent*
- Tipologías: *white, red, sparkling*

*Ahora ya hemos acabado con el preprocesamiento de este dataset, ahora ya podemos trabajar en el sin ningun problema, te invito a ver el resto de documentos donde seguire explicando los siguientes procesos, hasta ahora!*