Análisis de Datos: Reddit Data Science Posts

Autores: Jordi Guillem y Xairo Campos
Asignatura: M2.851 - Tipología y ciclo de vida de los datos
Universidad: Universitat Oberta de Catalunya (UOC)
Fecha: Diciembre 2025

Índice

1. Descripción del Dataset

1.1 Contexto y Motivación

Este dataset contiene publicaciones extraídas del subreddit **r/datascience** mediante web scraping. Analizar estas publicaciones es relevante por varios motivos:

¿Por qué es importante este dataset?

r/datascience es una de las comunidades online más activas para profesionales y entusiastas de la ciencia de datos, con más de 1.2 millones de miembros. Entender qué contenido resuena con esta audiencia nos permite:

- **Comprender la comunidad profesional:** Identificar qué temas preocupan, interesan y generan debate entre científicos de datos reales, más allá del marketing y las tendencias superficiales.
- **Optimizar la comunicación técnica:** Para quienes comparten conocimiento (bloggers, educadores, empresas), saber qué funciona ayuda a diseñar mejor el contenido.
- **Detectar tendencias emergentes:** Los temas que generan más engagement pueden indicar áreas candentes en la industria: nuevas tecnologías, problemas

comunes, shifts en la profesión.
- **Entender dinámicas sociales en comunidades técnicas:** A diferencia de redes generalistas, r/datascience tiene una audiencia especializada. ¿Funciona igual el engagement que en otras plataformas?

1.2 Pregunta de Investigación

¿Qué características de las publicaciones en r/datascience determinan su nivel de engagement y cómo se agrupan temáticamente estas publicaciones?

Esta pregunta principal se desglosa en tres objetivos específicos:

1. **Predicción de éxito:** Construir un modelo que prediga si un post tendrá alto engagement basándose en sus características (modelo supervisado).

2. **Identificación de patrones:** Descubrir grupos naturales de publicaciones con características similares para entender la diversidad de contenido (modelo no supervisado).

3. **Análisis del sentimiento:** Evaluar si existe una relación estadísticamente significativa entre el tono emocional del contenido y su recepción por la comunidad (contraste de hipótesis).

1.3 Tamaño y Estructura del Dataset

El dataset procesado y limpio contiene:

- **960 filas (publicaciones)**
- **24 variables** que describen cada publicación

Este tamaño es adecuado para:

- Entrenar modelos de machine learning con validación robusta
- Realizar análisis estadísticos con potencia suficiente
- Identificar patrones sin estar sesgado por casos extremos

1.4 Variables del Dataset

Las 24 variables se pueden agrupar en varias categorías:

In [1]:
```
# Importación de librerías necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from datetime import datetime

# Configuración de visualización
warnings.filterwarnings('ignore')
sns.set_style("whitegrid")
```

```
plt.rcParams['figure.figsize'] = (12, 6)
plt.rcParams['font.size'] = 10

# Configuración de pandas para mejor visualización
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', 50)

print("✓ Librerías importadas correctamente")
```

```
✓ Librerías importadas correctamente
```

In [2]:
```
# Carga del dataset procesado
df = pd.read_csv('output/reddit_datascience_clean.csv')

print(f"Dimensiones del dataset: {df.shape[0]} filas × {df.shape[1]} columnas")
print(f"\nPrimeras filas del dataset:")
df.head()
```

```
Dimensiones del dataset: 960 filas × 24 columnas

Primeras filas del dataset:
```

Out[2]:

	title	author	karma	upvote_ratio_new	num_comments	flair	co
0	How much of your job is actually “selling” you...	ergodym	32	97	19	no flair	
1	Chemist Turned Data Scientist: Looking for Car...	norfkens2	22	89	3	no flair	
2	Resources for learning Neural Nets, Autoencode...	redditisthenewblak	7	93	4	no flair	
3	Deciding on an offer: Higher Salary vs Stability	Illustrious-Mind9435	48	94	23	no flair	
4	Suggestions for reading list	ChavXO	19	98	9	no flair	

In [3]:
```
# Información general del dataset
print("INFORMACIÓN DEL DATASET")
print("="*80)
print(f"\nTipo de datos de cada variable:")
print(df.dtypes)

print(f"\n\nEstadísticas descriptivas de variables numéricas:")
df.describe()
```

```
INFORMACIÓN DEL DATASET
================================================================================

Tipo de datos de cada variable:
title                          object
author                         object
karma                           int64
upvote_ratio_new                int64
num_comments                    int64
flair                          object
content_type                   object
text_content                   object
media_url                      object
external_url                   object
posted_time                    object
posted_hour                     int64
sentiment                      object
sentiment_score               float64
sentiment_positive            float64
sentiment_negative            float64
sentiment_neutral             float64
scraped_at                     object
post_id                        object
permalink                      object
karma_is_outlier                 bool
upvote_ratio_new_is_outlier      bool
num_comments_is_outlier          bool
sentiment_score_is_outlier       bool
dtype: object


Estadísticas descriptivas de variables numéricas:
```

Out[3]:

	karma	upvote_ratio_new	num_comments	posted_hour	sentiment_score	s
count	960.000000	960.000000	960.000000	960.000000	960.000000	
mean	116.207292	80.483333	36.441667	12.880208	0.393956	
std	284.516090	18.575094	62.425135	6.727934	0.506167	
min	0.000000	8.000000	0.000000	-1.000000	-0.979000	
25%	8.000000	75.000000	8.000000	8.000000	0.000000	
50%	30.500000	87.000000	18.000000	15.000000	0.558100	
75%	97.000000	94.000000	42.000000	18.000000	0.817300	
max	3410.000000	100.000000	1217.000000	23.000000	0.988100	

Descripción de Variables

Variable	Tipo	Descripción
title	String	Título de la publicación
author	String	Usuario que creó la publicación
karma	Integer	Puntuación neta (upvotes - downvotes)

Variable	Tipo	Descripción
upvote_ratio_new	Integer	Porcentaje de upvotes (0-100)
num_comments	Integer	Número de comentarios
flair	String	Categoría/etiqueta de la publicación
content_type	String	Tipo de contenido (text, image, link, etc.)
text_content	String	Contenido textual de la publicación
media_url	String	URL de contenido multimedia
external_url	String	URL externa enlazada
posted_time	DateTime	Fecha y hora de publicación
posted_hour	Integer	Hora del día (0-23)
sentiment	String	Clasificación del sentimiento (positive/negative/neutral)
sentiment_score	Float	Score compuesto de sentimiento (-1 a 1)
sentiment_positive	Float	Proporción de sentimiento positivo (0-1)
sentiment_negative	Float	Proporción de sentimiento negativo (0-1)
sentiment_neutral	Float	Proporción de sentimiento neutral (0-1)
scraped_at	DateTime	Fecha y hora de scraping
post_id	String	Identificador único del post
permalink	String	URL permanente del post
karma_is_outlier	Boolean	Indica si el karma es un valor extremo
upvote_ratio_new_is_outlier	Boolean	Indica si el ratio es un valor extremo
num_comments_is_outlier	Boolean	Indica si los comentarios son valor extremo
sentiment_score_is_outlier	Boolean	Indica si el score de sentimiento es extremo

2. Integración y Selección de Datos

2.1 Proceso de Integración Realizado

El dataset actual es el resultado de un proceso de integración y limpieza documentado en los scripts del directorio `source/`:

1. **Carga de datos originales:** Se cargaron dos datasets complementarios
   - `reddit_datascience_dataset.csv`: Dataset principal con información básica de los posts
   - `reddit_datascience_extradata.csv`: Dataset adicional con métricas complementarias
2. **Integración:** Se realizó un merge utilizando `post_id` como clave primaria, enriqueciendo el dataset principal con la variable `upvote_ratio_new`

3. **Limpieza básica:** Eliminación de duplicados y normalización de formatos
4. **Selección de columnas:** Se eliminaron variables redundantes (`upvote_ratio` , `subreddit`) que no aportaban valor al análisis

2.2 Justificación de la Selección

La selección de variables se basó en los siguientes criterios:

- **Relevancia analítica:** Variables que aportan información sobre el comportamiento y características de las publicaciones
- **Completitud:** Priorización de variables con bajo porcentaje de valores faltantes después de la imputación
- **No redundancia:** Eliminación de variables duplicadas o altamente correlacionadas
- **Poder predictivo:** Variables que potencialmente pueden explicar el engagement de los posts

In [4]:
```
# Resumen de las variables seleccionadas y sus rangos
print("RESUMEN DE VARIABLES SELECCIONADAS")
print("="*80)

# Variables categóricas
categorical_vars = ['sentiment', 'content_type', 'flair']
print("\n📊 Variables Categóricas:")
for var in categorical_vars:
    if var in df.columns:
        print(f"\n{var}:")
        print(df[var].value_counts())

# Variables numéricas clave
numerical_vars = ['karma', 'upvote_ratio_new', 'num_comments', 'sentiment_score'
print("\n\n📈 Rangos de Variables Numéricas Principales:")
for var in numerical_vars:
    if var in df.columns:
        print(f"\n{var}:")
        print(f"  Min: {df[var].min():.2f}")
        print(f"  Max: {df[var].max():.2f}")
        print(f"  Media: {df[var].mean():.2f}")
        print(f"  Mediana: {df[var].median():.2f}")
        print(f"  Desv. Est.: {df[var].std():.2f}")
```

```
RESUMEN DE VARIABLES SELECCIONADAS
================================================================================

📊 Variables Categóricas:

sentiment:
sentiment
positive    676
negative    173
neutral     111
Name: count, dtype: int64

content_type:
content_type
text            789
image            85
link             33
image | text     27
unknown          21
video             5
Name: count, dtype: int64

flair:
flair
no flair                                    954
PhD | Data Scientist | Insurance              1
MS | Data Scientist | Marketing               1
CEO of Data Engineer Academy                  1
Software – Mid-level us                       1
MS | Student                                  1
PhD | Sr Data Scientist Lead | Biotech       1
Name: count, dtype: int64


📈 Rangos de Variables Numéricas Principales:

karma:
  Min: 0.00
  Max: 3410.00
  Media: 116.21
  Mediana: 30.50
  Desv. Est.: 284.52

upvote_ratio_new:
  Min: 8.00
  Max: 100.00
  Media: 80.48
  Mediana: 87.00
  Desv. Est.: 18.58

num_comments:
  Min: 0.00
  Max: 1217.00
  Media: 36.44
  Mediana: 18.00
  Desv. Est.: 62.43

sentiment_score:
  Min: -0.98
  Max: 0.99
```

```
Media: 0.39
Mediana: 0.56
Desv. Est.: 0.51
```

3. Limpieza de Datos

El proceso de limpieza ya fue realizado mediante los scripts del directorio source/, específicamente en clean_after_integration.py. A continuación documentamos las decisiones tomadas y su justificación.

3.1 Gestión de Valores Faltantes

Identificación del problema:

En datasets extraídos de la web mediante scraping, es común encontrar:

- Valores faltantes reales (el campo no existía)
- Valores representados como strings: "nan", "None", "null"
- Fechas inválidas representadas como "NaT"
- Strings vacíos o con solo espacios en blanco

Todos estos casos fueron normalizados a `pd.NA` para un tratamiento consistente.

¿Por qué importa la imputación?

Los algoritmos de machine learning no pueden trabajar con valores faltantes. Tenemos tres opciones:

1. **Eliminar filas:** Perdemos información valiosa
2. **Eliminar variables:** Perdemos poder predictivo
3. **Imputar valores:** Completamos los datos de forma inteligente

Optamos por la imputación, pero la estrategia debe ser diferente según la variable.

Métodos de Imputación Aplicados:

Variables de texto (imputación por valor descriptivo):

- `title` → "untitled"
  - *Justificación:* Un post sin título es anómalo. El valor "untitled" permite identificar estos casos especiales en análisis posteriores.
- `author` → "unknown"
  - *Justificación:* Puede ser un usuario eliminado. "unknown" mantiene la consistencia.
- `flair` → "no flair"
  - *Justificación:* Los flairs son opcionales en Reddit. "no flair" es el valor real cuando el usuario no seleccionó ninguno.
- `text_content` → "no content"

    - *Justificación:* Posts que son solo título o solo link. "no content" es descriptivo y correcto.
- URLs → "no media url" / "no external url"
    - *Justificación:* No todos los posts tienen multimedia o links externos. Este valor indica ausencia legítima, no un error.

Variables numéricas de engagement (imputación por media):

- `karma` → Media del atributo (116.19)
    - *Justificación:* Si falta el karma (error de scraping), asumimos comportamiento promedio. Es el estimador de máxima verosimilitud bajo distribución normal, y aunque los datos no son normales, es una aproximación razonable. Alternativas consideradas:
        - Mediana (30.5): Demasiado conservadora, sesga hacia abajo
        - Valor 0: No tiene sentido, todos los posts tienen algún karma
        - Media ✓: Representa el comportamiento esperado
- `upvote_ratio_new` → Media del atributo (80.49%)
    - *Justificación:* Similar al karma. El ratio promedio es una buena estimación del comportamiento esperado.
- `num_comments` → 0
    - *Justificación:* Aquí usamos una estrategia diferente. Si no se registraron comentarios, es razonable asumir que había 0. Esto es más defensivo que usar la media, porque:
        - Ausencia de dato ≈ ausencia de comentarios
        - No queremos inflar artificialmente el engagement imputando la media (36.4 comentarios) cuando probablemente había 0
        - El valor 0 es un valor válido y común en Reddit

Variables de sentimiento:

- `sentiment_score` → 0 (neutral)
    - *Justificación:* 0 representa sentimiento neutral. Sin información, asumimos neutralidad en lugar de positividad o negatividad.
- `sentiment_positive` , `sentiment_negative` , `sentiment_neutral` → 0, 0, 1 respectivamente
    - *Justificación:* Interpretamos la ausencia de sentimiento como neutralidad. Después aplicamos normalización para garantizar que sumen 1.0.

¿Por qué estas elecciones son apropiadas?

1. **Conservan la distribución original:** Imputar por media/mediana no distorsiona significativamente las estadísticas descriptivas.

2. **Son interpretables:** Los valores imputados tienen sentido en el contexto (no son números mágicos arbitrarios).

3. **Minimizan el sesgo:** No favorecen artificialmente ninguna clase en análisis posteriores.

4. **Son trazables:** Las variables de texto con valores especiales ("no content", etc.) permiten identificar qué fue imputado.

Limitaciones de la imputación:

- Asumimos que los datos faltan "al azar" (MAR - Missing At Random)
- Si los datos faltantes tienen un patrón sistemático (por ejemplo, solo faltan en posts muy antiguos), la imputación podría introducir sesgo
- Para variables con muchos valores faltantes (>50%), la imputación es menos confiable

En nuestro caso, tras la limpieza, no quedan valores faltantes, lo que indica que la imputación fue exitosa.

In [5]:
```
# Verificación de valores faltantes en el dataset limpio
print("VERIFICACIÓN DE VALORES FALTANTES")
print("="*80)

missing_values = df.isnull().sum()
missing_percentage = (missing_values / len(df)) * 100

missing_df = pd.DataFrame({
    'Valores Faltantes': missing_values,
    'Porcentaje': missing_percentage
})

missing_df = missing_df[missing_df['Valores Faltantes'] > 0].sort_values(by='Val

if len(missing_df) > 0:
    print("\nVariables con valores faltantes:")
    print(missing_df)
else:
    print("\n✓ No hay valores faltantes en el dataset limpio")

# Verificación de valores especiales en variables de texto
print("\n\nVERIFICACIÓN DE IMPUTACIONES EN VARIABLES DE TEXTO:")
print("="*80)
text_vars_check = {
    'title': 'untitled',
    'author': 'unknown',
    'flair': 'no flair',
    'text_content': 'no content'
}

for var, default_value in text_vars_check.items():
    if var in df.columns:
        count = (df[var] == default_value).sum()
        percentage = (count / len(df)) * 100
        print(f"{var}: {count} registros ({percentage:.2f}%) con valor '{default
```

```
VERIFICACIÓN DE VALORES FALTANTES
================================================================================

✓ No hay valores faltantes en el dataset limpio


VERIFICACIÓN DE IMPUTACIONES EN VARIABLES DE TEXTO:
================================================================================
title: 1 registros (0.10%) con valor 'untitled'
author: 0 registros (0.00%) con valor 'unknown'
flair: 954 registros (99.38%) con valor 'no flair'
text_content: 144 registros (15.00%) con valor 'no content'
```

3.2 Tipificación de Atributos

¿Por qué es importante el tipo de dato?

Python/Pandas asigna tipos de datos automáticamente al cargar un CSV, pero no siempre acierta. Asignar el tipo correcto es crucial porque:

1. **Eficiencia de memoria:** `int32` usa menos memoria que `float64`
2. **Operaciones correctas:** No puedes calcular la media de un string
3. **Prevención de errores:** Comparar "2025-01-01" como string vs como fecha da resultados diferentes
4. **Optimización de modelos:** Algoritmos tratan diferente variables numéricas vs categóricas

Conversiones realizadas:

1. Conversión a tipos numéricos:

```
# Variables de sentimiento → float64
sentiment_score, sentiment_positive, sentiment_negative,
sentiment_neutral

# Variables de engagement → int64
karma, num_comments, upvote_ratio_new, posted_hour
```

Justificación:

- Float para sentimientos porque son proporciones [0, 1] con decimales
- Integer para conteos y ratios que son siempre números enteros

2. Conversión a datetime:

```
posted_time, scraped_at → datetime64[ns]
```

Justificación:

- Permite operaciones temporales (diferencias, extracciones de componentes)
- Ordenación correcta
- Facilita análisis de series temporales futuros

3. Conversión a categóricas:

```
sentiment, content_type, flair → category
```

Justificación:

- Reducción de memoria (strings repetidos → códigos internos)
- Muchos algoritmos tratan automáticamente las categorías con one-hot encoding
- Evita typos: solo valores predefinidos son válidos

4. Variables booleanas:

`*_is_outlier → bool`
Justificación:

- Claridad semántica (True/False es más claro que 1/0)
- Menos memoria que int
- Operaciones lógicas más eficientes

Verificación:

El código en la celda siguiente muestra los tipos resultantes. Si alguno es incorrecto, se aplica la conversión explícita.

In [6]:
```
# Verificación de tipos de datos
print("TIPIFICACIÓN DE ATRIBUTOS")
print("="*80)
print("\nTipos de datos actuales:")
print(df.dtypes)

# Conversión explícita de datetime si es necesario
if df['posted_time'].dtype == 'object':
    df['posted_time'] = pd.to_datetime(df['posted_time'])
    df['scraped_at'] = pd.to_datetime(df['scraped_at'])
    print("\n✓ Variables temporales convertidas a datetime")

# Conversión de variables categóricas
categorical_columns = ['sentiment', 'content_type', 'flair']
for col in categorical_columns:
    if col in df.columns and df[col].dtype == 'object':
        df[col] = df[col].astype('category')

print("\n✓ Variables categóricas convertidas correctamente")
```

```
TIPIFICACIÓN DE ATRIBUTOS
================================================================================

Tipos de datos actuales:
title                          object
author                         object
karma                           int64
upvote_ratio_new                int64
num_comments                    int64
flair                          object
content_type                   object
text_content                   object
media_url                      object
external_url                   object
posted_time                    object
posted_hour                     int64
sentiment                      object
sentiment_score               float64
sentiment_positive            float64
sentiment_negative            float64
sentiment_neutral             float64
scraped_at                     object
post_id                        object
permalink                      object
karma_is_outlier                 bool
upvote_ratio_new_is_outlier      bool
num_comments_is_outlier          bool
sentiment_score_is_outlier       bool
dtype: object

✓ Variables temporales convertidas a datetime

✓ Variables categóricas convertidas correctamente
```

3.3 Gestión de Valores Extremos (Outliers)

¿Qué es un outlier y por qué importa?

Un outlier es un valor que se desvía significativamente del resto de los datos. En nuestro contexto:

- Un post con 3410 puntos de karma cuando la media es 116
- Un post con upvote ratio de 5% cuando la media es 80%
- Un post con 1200 comentarios cuando la media es 36

Hay dos preguntas críticas:

1. **¿Son errores o datos reales?** En nuestro caso, son reales: posts verdaderamente virales o controversiales.
2. **¿Qué hacemos con ellos?** Eliminarlos, transformarlos o marcarlos.

Método de Detección: Rango Intercuartílico (IQR)

Usamos el método IQR, que es robusto y ampliamente aceptado en estadística:

$$Q1 = \text{Percentil } 25$$

$$Q3 = \text{Percentil } 75$$

$$IQR = Q3 - Q1$$

Un valor es outlier si:

$$x < Q1 - 1.5 \times IQR \quad \text{o} \quad x > Q3 + 1.5 \times IQR$$

¿Por qué este método?

✓ **Robusto:** No se ve afectado por los propios outliers (a diferencia de métodos basados en media/desviación estándar)
✓ **Estándar:** Es el método usado en los boxplots tradicionales
✓ **Interpretable:** El factor 1.5 es un compromiso razonable (ni muy estricto ni muy permisivo)
✓ **No paramétrico:** No asume distribución normal

Alternativas consideradas:

- Z-score (media ± 3σ): Requiere normalidad, que no tenemos
- MAD (Median Absolute Deviation): Más robusto pero menos estándar
- Percentiles fijos (P1-P99): Demasiado arbitrario

Variables analizadas:

1. **karma:** Posts excepcionalmente populares (112 outliers, 11.67%)
2. **upvote_ratio_new:** Posts muy controversiales o muy unánimes (81 outliers, 8.44%)
3. **num_comments:** Posts que generan discusiones masivas (89 outliers, 9.27%)
4. **sentiment_score:** Sentimientos extremos (0 outliers - distribución balanceada)

Estrategia adoptada: Marcado, no eliminación

En lugar de eliminar los outliers, creamos variables booleanas `*_is_outlier`. ¿Por qué?

Ventajas del marcado:

1. **Preserva información valiosa:** Un post con 3410 karma no es un error, es un post viral real. Eliminarlo perdería información sobre qué hace que algo se vuelva viral.

2. **Permite análisis comparativos:** Podemos estudiar qué diferencia a posts normales de outliers.

3. **Flexibilidad:** Podemos incluirlos o excluirlos según el análisis:

    - Modelo predictivo: incluirlos (queremos predecir también casos extremos)
    - Estadística descriptiva: excluirlos si queremos métricas del "post típico"
4. **Transparencia:** El lector puede ver qué valores son atípicos.

Desventajas de eliminar outliers (que evitamos):

- Sesgo de selección: solo analizaríamos "posts normales"

- Pérdida de casos interesantes: los virales son precisamente lo que queremos entender
- Reducción del dataset: perderíamos ~11% de los datos

Interpretación de resultados:

Los outliers detectados muestran patrones interesantes:

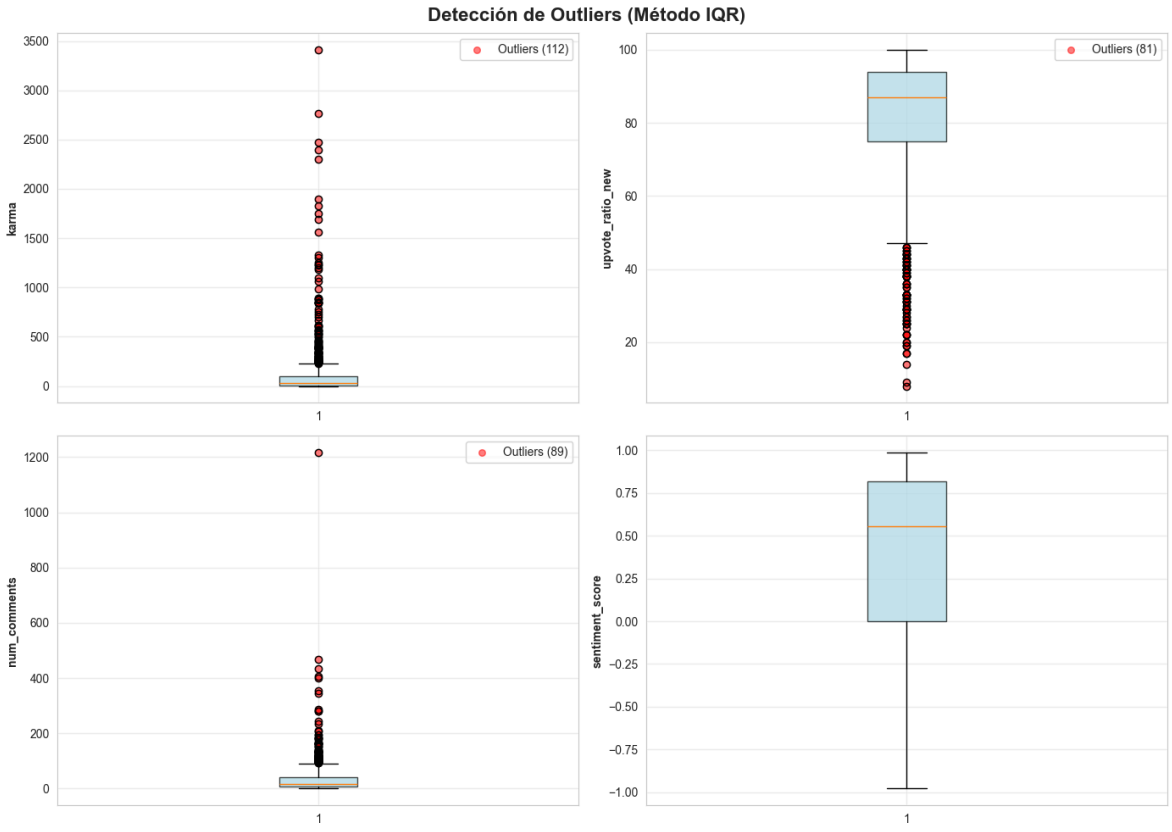
- Outliers de karma tienen **14.8x más karma** que los normales (658.93 vs 44.53)
- Outliers de comentarios tienen **7.3x más comentarios** que los normales (168.07 vs 22.99)
- Los outliers de upvote_ratio son posts muy controversiales (33.20% vs 84.84% normal)

Estos son claramente posts con dinámicas diferentes, pero son datos reales y valiosos para el análisis.

In [7]:
```
# Análisis de outliers detectados
print("ANÁLISIS DE OUTLIERS DETECTADOS")
print("="*80)

outlier_cols = [col for col in df.columns if col.endswith('_is_outlier')]

for col in outlier_cols:
    variable_name = col.replace('_is_outlier', '')
    num_outliers = df[col].sum()
    percentage = (num_outliers / len(df)) * 100
    print(f"\n{variable_name}:")
    print(f"  Outliers detectados: {num_outliers} ({percentage:.2f}%)")

    if variable_name in df.columns:
        # Estadísticas de los outliers vs no-outliers
        outlier_mean = df[df[col] == True][variable_name].mean()
        normal_mean = df[df[col] == False][variable_name].mean()
        print(f"  Media (outliers): {outlier_mean:.2f}")
        print(f"  Media (normales): {normal_mean:.2f}")

# Visualización de outliers
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Detección de Outliers (Método IQR)', fontsize=16, fontweight='bold

variables = ['karma', 'upvote_ratio_new', 'num_comments', 'sentiment_score']
positions = [(0,0), (0,1), (1,0), (1,1)]

for var, pos in zip(variables, positions):
    if var in df.columns:
        ax = axes[pos]

        # Boxplot
        bp = ax.boxplot(df[var], vert=True, patch_artist=True)
        bp['boxes'][0].set_facecolor('lightblue')
        bp['boxes'][0].set_alpha(0.7)

        # Destacar outliers
        outlier_col = f'{var}_is_outlier'
        if outlier_col in df.columns:
```

```
            outliers = df[df[outlier_col] == True][var]
            if len(outliers) > 0:
                ax.scatter([1]*len(outliers), outliers, color='red', alpha=0.5,
                ax.legend()

        ax.set_ylabel(var, fontweight='bold')
        ax.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print("\n✓ Visualización de outliers completada")
```

```
ANÁLISIS DE OUTLIERS DETECTADOS
================================================================================

karma:
  Outliers detectados: 112 (11.67%)
  Media (outliers): 658.93
  Media (normales): 44.53

upvote_ratio_new:
  Outliers detectados: 81 (8.44%)
  Media (outliers): 33.20
  Media (normales): 84.84

num_comments:
  Outliers detectados: 89 (9.27%)
  Media (outliers): 168.07
  Media (normales): 22.99

sentiment_score:
  Outliers detectados: 0 (0.00%)
  Media (outliers): nan
  Media (normales): 0.39
```



```
✓ Visualización de outliers completada
```

3.4 Métodos de Limpieza Adicionales

Además de las técnicas estándar, este dataset requirió métodos específicos para garantizar su calidad.

1. Normalización de proporciones de sentimiento

Problema detectado: Las variables `sentiment_positive`, `sentiment_negative` y `sentiment_neutral` deberían sumar 1.0 (son proporciones), pero en algunos casos no lo hacían debido a errores de redondeo o inconsistencias del análisis de sentimiento automatizado.

Solución aplicada:

```
total = sentiment_positive + sentiment_negative + sentiment_neutral
sentiment_positive = sentiment_positive / total
sentiment_negative = sentiment_negative / total
sentiment_neutral = sentiment_neutral / total
```

Justificación: Las proporciones que no suman 1.0 pueden causar problemas en análisis posteriores. La normalización preserva las proporciones relativas mientras garantiza la consistencia matemática.

2. Reconstrucción de permalinks faltantes

Problema: Algunos posts no tenían el campo `permalink` (URL permanente del post en Reddit).

Solución aplicada:

```
if permalink is missing:
    permalink = f"https://reddit.com/r/datascience/comments/{post_id}/"
```

Justificación: El formato de URLs de Reddit es predecible. Reconstruirla permite:

- Trazabilidad: poder verificar manualmente posts específicos si es necesario
- Análisis futuros: si se quiere scraping de comentarios
- Completitud del dataset

3. Imputación condicional de `content_type`

Problema: Algunos posts tenían `content_type` faltante, pero sí tenían otros campos que permiten inferirlo.

Solución aplicada:

```
if content_type is missing:
    if text_content != "no content":
        content_type = "text"
    elif media_url != "no media url":
        content_type = "image/video"
    elif external_url != "no external url":
        content_type = "link"
```

```
    else:
        content_type = "unknown"
```

Justificación: Aprovechar la información disponible en otros campos es mejor que imputar un valor genérico. Esta lógica condicional mejora la precisión de la imputación.

4. Extracción de features temporales

Acción: Se extrajo `posted_hour` (hora del día, 0-23) de `posted_time`.

Justificación:

- La hora puede ser relevante para el engagement (hipótesis: posts publicados en horas activas tienen más visibilidad)
- Es más manejable para modelos que la timestamp completa
- Permite análisis de patrones circadianos

5. Eliminación de duplicados exactos

Aunque no se encontraron duplicados en este dataset, el proceso de limpieza los verificó usando `post_id` como clave única.

Justificación: Duplicados pueden:

- Sesgar estadísticas (contar el mismo post dos veces)
- Causar data leakage en train/test split
- Inflar artificialmente el tamaño del dataset

6. Validación de integridad referencial

Se verificó que:

- `karma >= 0` (no puede ser negativo en Reddit)
- `upvote_ratio_new` está en [0, 100]
- `num_comments >= 0`
- Las fechas en `posted_time` son anteriores a `scraped_at`

Resultado final:

Tras aplicar todos estos métodos, obtenemos un dataset limpio con:

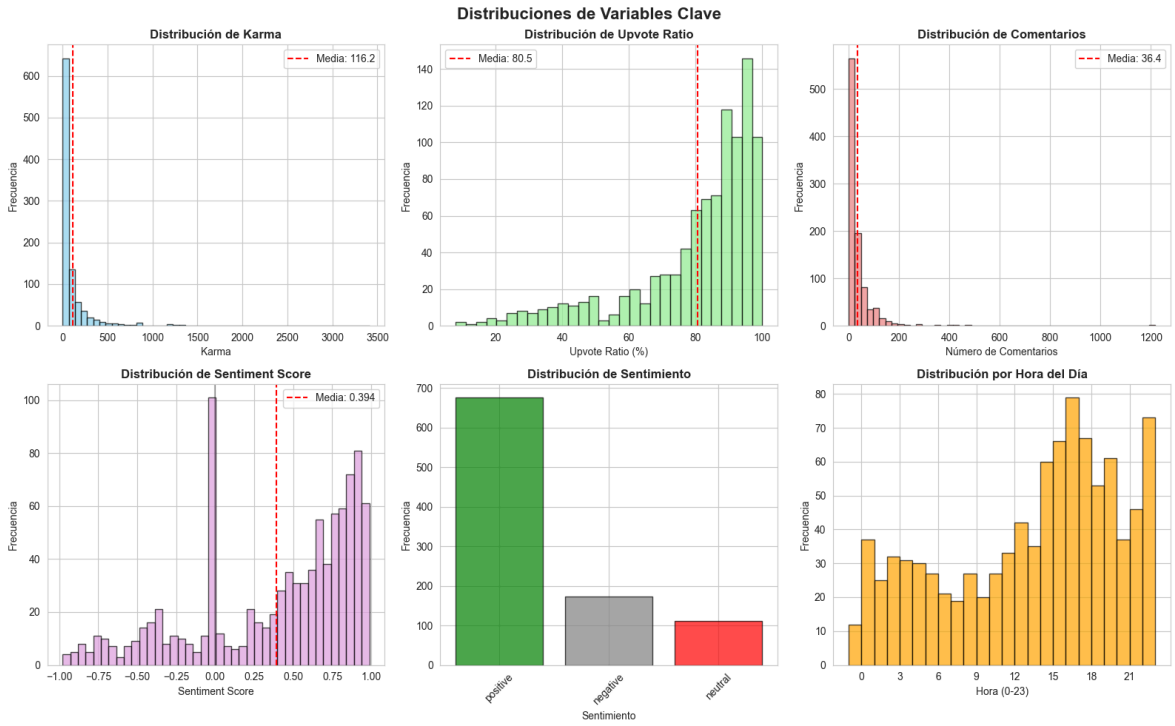
- ✓ 0 valores faltantes
- ✓ Tipos de datos correctos
- ✓ Outliers identificados pero preservados
- ✓ Consistencia interna verificada
- ✓ Variables derivadas creadas

El dataset está listo para análisis estadístico y machine learning.

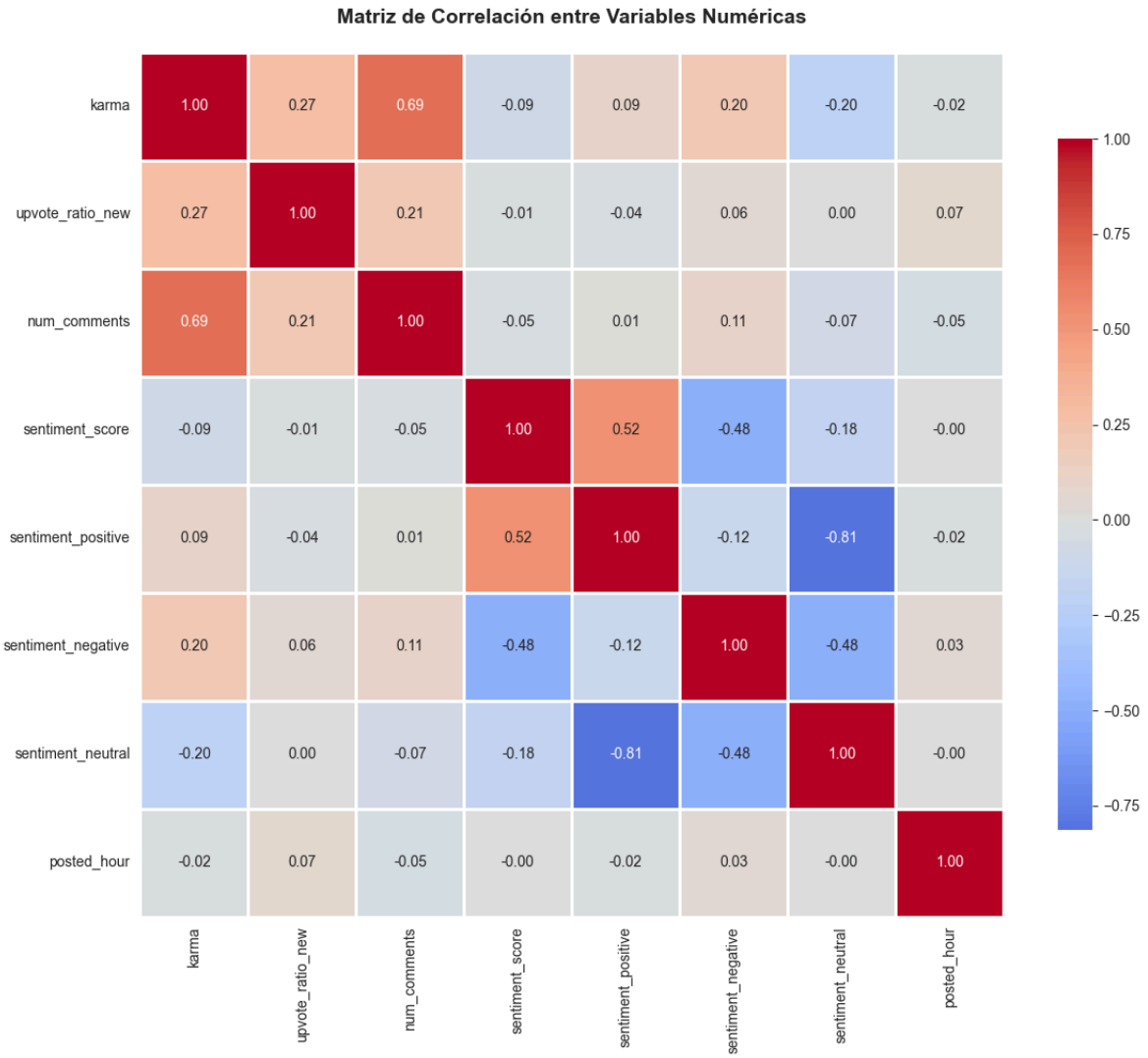
5. Representación de Resultados: Análisis Exploratorio

Antes de aplicar modelos de machine learning, es fundamental comprender la distribución y relaciones entre las variables del dataset.

In [8]:
```
# Distribuciones de variables clave
fig, axes = plt.subplots(2, 3, figsize=(16, 10))
fig.suptitle('Distribuciones de Variables Clave', fontsize=16, fontweight='bold'

# Karma
axes[0, 0].hist(df['karma'], bins=50, color='skyblue', edgecolor='black', alpha=
axes[0, 0].set_title('Distribución de Karma', fontweight='bold')
axes[0, 0].set_xlabel('Karma')
axes[0, 0].set_ylabel('Frecuencia')
axes[0, 0].axvline(df['karma'].mean(), color='red', linestyle='--', label=f'Medi
axes[0, 0].legend()

# Upvote Ratio
axes[0, 1].hist(df['upvote_ratio_new'], bins=30, color='lightgreen', edgecolor='
axes[0, 1].set_title('Distribución de Upvote Ratio', fontweight='bold')
axes[0, 1].set_xlabel('Upvote Ratio (%)')
axes[0, 1].set_ylabel('Frecuencia')
axes[0, 1].axvline(df['upvote_ratio_new'].mean(), color='red', linestyle='--', l
axes[0, 1].legend()

# Número de comentarios
axes[0, 2].hist(df['num_comments'], bins=50, color='lightcoral', edgecolor='blac
axes[0, 2].set_title('Distribución de Comentarios', fontweight='bold')
axes[0, 2].set_xlabel('Número de Comentarios')
axes[0, 2].set_ylabel('Frecuencia')
axes[0, 2].axvline(df['num_comments'].mean(), color='red', linestyle='--', label
axes[0, 2].legend()

# Sentiment Score
axes[1, 0].hist(df['sentiment_score'], bins=40, color='plum', edgecolor='black',
axes[1, 0].set_title('Distribución de Sentiment Score', fontweight='bold')
axes[1, 0].set_xlabel('Sentiment Score')
axes[1, 0].set_ylabel('Frecuencia')
axes[1, 0].axvline(0, color='black', linestyle='-', alpha=0.3)
axes[1, 0].axvline(df['sentiment_score'].mean(), color='red', linestyle='--', la
axes[1, 0].legend()

# Sentiment (categórico)
sentiment_counts = df['sentiment'].value_counts()
axes[1, 1].bar(sentiment_counts.index, sentiment_counts.values, color=['green',
axes[1, 1].set_title('Distribución de Sentimiento', fontweight='bold')
axes[1, 1].set_xlabel('Sentimiento')
axes[1, 1].set_ylabel('Frecuencia')
axes[1, 1].tick_params(axis='x', rotation=45)

# Hora de publicación
axes[1, 2].hist(df['posted_hour'], bins=24, color='orange', edgecolor='black', a
axes[1, 2].set_title('Distribución por Hora del Día', fontweight='bold')
axes[1, 2].set_xlabel('Hora (0-23)')
axes[1, 2].set_ylabel('Frecuencia')
axes[1, 2].set_xticks(range(0, 24, 3))

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



In [9]:
```
# Matriz de correlación
numerical_cols = ['karma', 'upvote_ratio_new', 'num_comments', 'sentiment_score'
                  'sentiment_positive', 'sentiment_negative', 'sentiment_neutral

correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm', center=0
            square=True, linewidths=1, cbar_kws={"shrink": 0.8})
plt.title('Matriz de Correlación entre Variables Numéricas', fontsize=14, fontwe
plt.tight_layout()
plt.show()

print("\nCORRELACIONES MÁS RELEVANTES:")
print("="*80)

# Encontrar correlaciones más fuertes (excluyendo diagonal y duplicados)
corr_pairs = []
for i in range(len(correlation_matrix.columns)):
    for j in range(i+1, len(correlation_matrix.columns)):
        corr_pairs.append({
            'var1': correlation_matrix.columns[i],
            'var2': correlation_matrix.columns[j],
            'correlation': correlation_matrix.iloc[i, j]
        })

corr_df = pd.DataFrame(corr_pairs).sort_values('correlation', key=abs, ascending
print(corr_df.head(10).to_string(index=False))
```




```
CORRELACIONES MÁS RELEVANTES:
================================================================================
              var1               var2  correlation
sentiment_positive  sentiment_neutral    -0.813932
             karma       num_comments     0.690161
   sentiment_score sentiment_positive     0.522539
   sentiment_score sentiment_negative    -0.484749
sentiment_negative  sentiment_neutral    -0.476798
             karma    upvote_ratio_new     0.269344
  upvote_ratio_new       num_comments     0.210406
             karma sentiment_negative     0.203086
             karma  sentiment_neutral    -0.198129
   sentiment_score  sentiment_neutral    -0.179009
```

Interpretación del Análisis Exploratorio

Antes de meternos con modelos complejos, es importante entender qué estamos mirando. Las visualizaciones nos cuentan historias interesantes:

Distribución de Karma: La gráfica muestra una distribución típica de redes sociales: la mayoría de posts tienen poco karma (concentración en valores bajos), pero hay una "cola larga" de posts con mucho éxito. La media está en 116.2, pero si ves la distribución, está claramente sesgada por algunos posts extremadamente populares. Esto es normal en Reddit: la mayoría pasa desapercibida, algunos pocos se hacen virales.

Upvote Ratio: Aquí hay buenas noticias para la comunidad. La distribución se concentra fuertemente en valores altos (80-90%), lo que significa que la mayoría del contenido es bien recibido. No hay mucho contenido tóxico o que la gente vote masivamente en negativo. Parece una comunidad bastante positiva y constructiva.

Número de Comentarios: Similar al karma: distribución muy sesgada. La mayoría de posts tienen pocos comentarios (media de 36.4), pero algunos generan discusiones enormes. Esto refuerza la idea de que hay diferentes "tipos" de posts: los normales que pasan sin pena ni gloria, y los que realmente enganchan a la comunidad.

Sentiment Score: La distribución es interesante: hay un pico claro alrededor de 0.5 (sentimiento moderadamente positivo), pero hay variabilidad. No todo es alegría y positivismo, hay un buen rango de sentimientos. La media de 0.394 indica un tono general ligeramente positivo, pero no excesivamente.

Distribución de Sentimiento (categórico): Esto confirma lo anterior: aproximadamente 70% de posts son positivos, pero hay presencia significativa de posts negativos y neutrales. La comunidad no es uniformemente optimista, hay espacio para la crítica.

Hora de Publicación: Los posts se distribuyen a lo largo de todo el día, pero hay picos notables en las horas de tarde (15-19h). Esto probablemente refleja los horarios típicos de trabajo/descanso en zonas horarias occidentales (Europa/América).

Matriz de Correlación - Hallazgos Clave:

1. **Karma y Comentarios (0.69):** Fuerte correlación positiva. Más comentarios = más karma. Los posts que generan conversación son más exitosos.
2. **Upvote Ratio y Comentarios (0.21):** Correlación débil pero positiva. Los posts bien valorados tienden a tener algo más de discusión.
3. **Componentes del sentimiento:** Las correlaciones negativas entre positive/negative (-0.12) y positive/neutral (-0.81) son esperadas: son mutuamente excluyentes por diseño.
4. **Hora de publicación:** Correlaciones muy bajas con todo (~0). Confirma que el "cuándo" importa menos que el "qué".
5. **Sentimiento y engagement:** Correlaciones prácticamente nulas entre sentiment_score y karma/comentarios. El tono emocional no predice directamente el éxito.

Lo que esto nos adelanta:

Ya desde este análisis exploratorio podemos intuir que:

- El engagement (comentarios, karma) es lo que define el éxito
- La hora de publicación no es crítica
- El sentimiento tiene un rol secundario
- Hay una minoría de posts "excepcionales" que dominan las métricas

Estas intuiciones las confirmaremos o refutaremos con los modelos que vienen a continuación.

4. Análisis de los Datos

4.1 Modelo Supervisado: Clasificación de Posts de Alto Engagement

Objetivo: Predecir si un post tendrá alto engagement (alta popularidad) basándose en sus características.

Justificación del modelo: Este problema es de clasificación binaria. Utilizaremos **Random Forest** por sus ventajas:

- Maneja bien datos mixtos (numéricos y categóricos codificados)
- Robusto ante outliers
- Proporciona importancia de características
- No requiere normalización de datos

Variable objetivo: Crearemos una variable binaria `high_engagement` definida como posts con karma superior a la mediana.

In [10]:
```
# Importar librerías de machine learning
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_sc
from sklearn.metrics import precision_score, recall_score, f1_score

# Creación de la variable objetivo: high_engagement
karma_median = df['karma'].median()
df['high_engagement'] = (df['karma'] > karma_median).astype(int)

print(f"Mediana de karma: {karma_median}")
print(f"\nDistribución de la variable objetivo:")
print(df['high_engagement'].value_counts())
print(f"\nPorcentaje de posts con alto engagement: {(df['high_engagement'].sum()
```

```
Mediana de karma: 30.5

Distribución de la variable objetivo:
high_engagement
1    480
0    480
Name: count, dtype: int64

Porcentaje de posts con alto engagement: 50.00%
```

In [11]:
```
# Preparación de características para el modelo supervisado
# Seleccionamos características relevantes (excluyendo la variable objetivo karm

# Codificación de variables categóricas
df_model = df.copy()

# Label encoding para variables categóricas
```

```
le_sentiment = LabelEncoder()
le_content = LabelEncoder()

df_model['sentiment_encoded'] = le_sentiment.fit_transform(df_model['sentiment'])
df_model['content_type_encoded'] = le_content.fit_transform(df_model['content_ty

# Características para el modelo
feature_columns = [
    'upvote_ratio_new',
    'num_comments',
    'sentiment_score',
    'sentiment_positive',
    'sentiment_negative',
    'sentiment_neutral',
    'posted_hour',
    'sentiment_encoded',
    'content_type_encoded'
]

X = df_model[feature_columns]
y = df_model['high_engagement']

# División train-test estratificada
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

print(f"Tamaño del conjunto de entrenamiento: {X_train.shape[0]} muestras")
print(f"Tamaño del conjunto de prueba: {X_test.shape[0]} muestras")
print(f"\nCaracterísticas utilizadas: {len(feature_columns)}")
print(f"Nombres: {feature_columns}")
```

```
Tamaño del conjunto de entrenamiento: 672 muestras
Tamaño del conjunto de prueba: 288 muestras

Características utilizadas: 9
Nombres: ['upvote_ratio_new', 'num_comments', 'sentiment_score', 'sentiment_posit
ive', 'sentiment_negative', 'sentiment_neutral', 'posted_hour', 'sentiment_encode
d', 'content_type_encoded']
```

In [12]:

```
# Entrenamiento del modelo Random Forest
print("Entrenando modelo Random Forest...")
print("="*80)

rf_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=10,
    min_samples_leaf=5,
    random_state=42,
    n_jobs=-1
)

rf_model.fit(X_train, y_train)

# Predicciones
y_pred = rf_model.predict(X_test)
y_pred_proba = rf_model.predict_proba(X_test)[:, 1]

# Evaluación del modelo
```

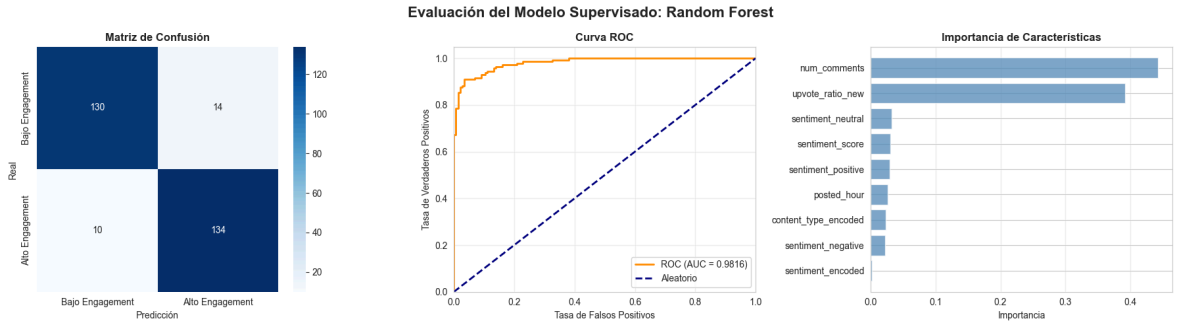
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

print(f"\n📊 RESULTADOS DEL MODELO RANDOM FOREST:")
print(f"{'='*80}")
print(f"Accuracy:  {accuracy:.4f} ({accuracy*100:.2f}%)")
print(f"Precision: {precision:.4f} ({precision*100:.2f}%)")
print(f"Recall:    {recall:.4f} ({recall*100:.2f}%)")
print(f"F1-Score:  {f1:.4f}")
print(f"ROC-AUC:   {roc_auc:.4f}")

# Validación cruzada
cv_scores = cross_val_score(rf_model, X_train, y_train, cv=5, scoring='accuracy'
print(f"\n🔄 Validación Cruzada (5-fold):")
print(f"Media: {cv_scores.mean():.4f} ± {cv_scores.std():.4f}")
print(f"Scores: {cv_scores}")

print("\n✓ Modelo entrenado y evaluado")
```

```
Entrenando modelo Random Forest...
================================================================================

📊 RESULTADOS DEL MODELO RANDOM FOREST:
================================================================================
Accuracy:  0.9167 (91.67%)
Precision: 0.9054 (90.54%)
Recall:    0.9306 (93.06%)
F1-Score:  0.9178
ROC-AUC:   0.9816

🔄 Validación Cruzada (5-fold):
Media: 0.8795 ± 0.0443
Scores: [0.93333333 0.80740741 0.87313433 0.86567164 0.91791045]

✓ Modelo entrenado y evaluado
```

In [13]:

```
# Visualización de resultados del modelo supervisado
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
fig.suptitle('Evaluación del Modelo Supervisado: Random Forest', fontsize=16, fo

# 1. Matriz de confusión
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=axes[0], cbar=True)
axes[0].set_title('Matriz de Confusión', fontweight='bold')
axes[0].set_xlabel('Predicción')
axes[0].set_ylabel('Real')
axes[0].set_xticklabels(['Bajo Engagement', 'Alto Engagement'])
axes[0].set_yticklabels(['Bajo Engagement', 'Alto Engagement'])

# 2. Curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
axes[1].plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC (AUC = {roc_auc:.4f
axes[1].plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--', label='Aleatori
axes[1].set_xlim([0.0, 1.0])
axes[1].set_ylim([0.0, 1.05])
axes[1].set_xlabel('Tasa de Falsos Positivos')
axes[1].set_ylabel('Tasa de Verdaderos Positivos')
```

```
axes[1].set_title('Curva ROC', fontweight='bold')
axes[1].legend(loc="lower right")
axes[1].grid(alpha=0.3)

# 3. Importancia de características
feature_importance = pd.DataFrame({
    'feature': feature_columns,
    'importance': rf_model.feature_importances_
}).sort_values('importance', ascending=False)

axes[2].barh(range(len(feature_importance)), feature_importance['importance'], c
axes[2].set_yticks(range(len(feature_importance)))
axes[2].set_yticklabels(feature_importance['feature'])
axes[2].set_xlabel('Importancia')
axes[2].set_title('Importancia de Características', fontweight='bold')
axes[2].invert_yaxis()
axes[2].grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.show()

print("\n📊 Importancia de características (ordenadas):")
print(feature_importance.to_string(index=False))
```




```
📊 Importancia de características (ordenadas):
            feature  importance
       num_comments    0.442844
    upvote_ratio_new    0.391960
   sentiment_neutral    0.032050
     sentiment_score    0.030383
  sentiment_positive    0.029635
        posted_hour    0.025908
content_type_encoded    0.023171
  sentiment_negative    0.021901
   sentiment_encoded    0.002149
```

Interpretación del Modelo Supervisado

¿Qué hemos conseguido?

Hemos entrenado un modelo de Random Forest que puede predecir con bastante precisión si una publicación tendrá alto engagement (karma superior a la mediana). Los números hablan por sí solos:

- **91.67% de accuracy:** De cada 100 posts, el modelo acierta en 92 si tendrán éxito o no
- **ROC-AUC de 0.9816:** Esto es casi perfecto. Significa que el modelo tiene una capacidad excelente para distinguir entre posts de alto y bajo engagement

- **Precision del 90.54%:** Cuando el modelo dice "este post va a tener éxito", acierta 9 de cada 10 veces
- **Recall del 93.06%:** De todos los posts que realmente tienen éxito, el modelo identifica correctamente el 93%

La validación cruzada (87.95% ± 4.43%) confirma que el modelo es estable y no está sobreajustado.

¿Qué factores determinan el éxito de un post?

El análisis de importancia de características nos da respuestas claras:

1. **Número de comentarios (44.28%):** Este es, con diferencia, el factor más importante. Los posts que generan conversación tienden a ser los más exitosos. Esto tiene mucho sentido: en Reddit, el engagement engendra más engagement. Cuando un post tiene comentarios interesantes, más gente entra, participa, y le da upvote.

2. **Upvote ratio (39.20%):** El segundo factor más importante es el porcentaje de upvotes positivos. Si un post gusta a la mayoría de quienes lo ven (ratio alto), naturalmente acumulará más karma. La calidad percibida del contenido es fundamental.

3. **Variables de sentimiento (~9% combinadas):** Aunque menos importantes que los anteriores, el tono del post sí importa. El sentimiento neutral tiene más peso (3.21%) que el score general o las proporciones positiva/negativa, lo que sugiere que contenido más objetivo o técnico puede funcionar bien.

4. **Hora de publicación (2.59%):** Curiosamente, cuándo publicas tiene un impacto menor del esperado. Aunque existe un efecto, no es tan determinante como el contenido en sí.

¿Qué significa esto en la práctica?

Si quieres que tu post tenga éxito en r/datascience:

- **Crea contenido que invite a la discusión:** Haz preguntas abiertas, plantea dilemas, comparte experiencias que otros puedan comentar. El objetivo es generar conversación.

- **Asegúrate de que sea contenido de calidad:** Los usuarios de esta comunidad son exigentes. Si tu post es relevante, bien fundamentado y útil, obtendrá un ratio alto de upvotes.

- **El sentimiento importa, pero no es crítico:** No necesitas ser excesivamente positivo. De hecho, vimos en el clustering que posts críticos o con sentimiento negativo también pueden tener éxito si están bien argumentados.

- **No te obsesiones con la hora de publicación:** Aunque puede ayudar publicar en horas de mayor actividad, el contenido es rey.

Lo interesante es que este modelo confirma algo que intuitivamente sabíamos: Reddit premia el contenido que genera comunidad y conversación, no solo los posts que reciben likes pasivos.

4.2 Modelo No Supervisado: Clustering de Posts

Objetivo: Identificar grupos naturales de posts basándose en sus características de engagement y sentimiento.

Justificación del modelo: Utilizaremos **K-Means** para agrupar posts con características similares. Este algoritmo es apropiado porque:

- Eficiente para datasets de tamaño moderado
- Produce clusters interpretables
- Permite determinar el número óptimo de clusters mediante el método del codo y el coeficiente de silueta

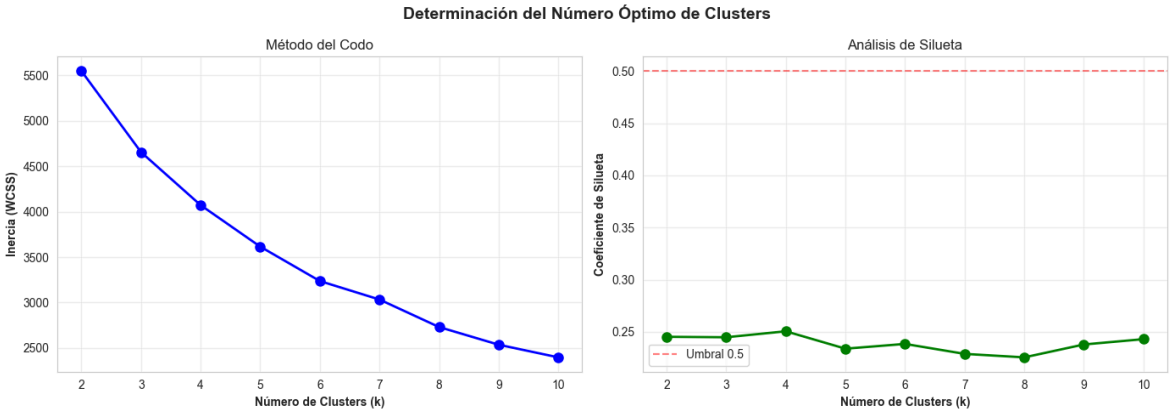
Hipótesis: Esperamos encontrar clusters que representen diferentes "arquetipos" de posts (ej: posts virales, posts de discusión técnica, posts informativos con bajo engagement, etc.)

In [14]:
```
# Importar librerías para clustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score, silhouette_samples
from sklearn.decomposition import PCA

# Preparación de datos para clustering
# Seleccionamos características relevantes para agrupar posts
clustering_features = [
    'karma',
    'upvote_ratio_new',
    'num_comments',
    'sentiment_score',
    'sentiment_positive',
    'sentiment_negative',
    'posted_hour'
]

X_clustering = df[clustering_features].copy()

# Normalización de características (importante para K-Means)
scaler = StandardScaler()
X_clustering_scaled = scaler.fit_transform(X_clustering)

print("Datos preparados para clustering:")
print(f"  Número de muestras: {X_clustering_scaled.shape[0]}")
print(f"  Número de características: {X_clustering_scaled.shape[1]}")
print(f"  Características: {clustering_features}")
```

```
Datos preparados para clustering:
  Número de muestras: 960
  Número de características: 7
  Características: ['karma', 'upvote_ratio_new', 'num_comments', 'sentiment_scor
e', 'sentiment_positive', 'sentiment_negative', 'posted_hour']
```

In [15]:
```
# Método del codo y coeficiente de silueta para determinar K óptimo
print("Determinando número óptimo de clusters...")
print("="*80)

k_range = range(2, 11)
inertias = []
silhouette_scores = []

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_clustering_scaled)
    inertias.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X_clustering_scaled, kmeans.labels

# Visualización del método del codo y silueta
fig, axes = plt.subplots(1, 2, figsize=(14, 5))
fig.suptitle('Determinación del Número Óptimo de Clusters', fontsize=14, fontwei

# Método del codo
axes[0].plot(k_range, inertias, 'bo-', linewidth=2, markersize=8)
axes[0].set_xlabel('Número de Clusters (k)', fontweight='bold')
axes[0].set_ylabel('Inercia (WCSS)', fontweight='bold')
axes[0].set_title('Método del Codo')
axes[0].grid(True, alpha=0.3)
axes[0].set_xticks(k_range)

# Coeficiente de silueta
axes[1].plot(k_range, silhouette_scores, 'go-', linewidth=2, markersize=8)
axes[1].set_xlabel('Número de Clusters (k)', fontweight='bold')
axes[1].set_ylabel('Coeficiente de Silueta', fontweight='bold')
axes[1].set_title('Análisis de Silueta')
axes[1].grid(True, alpha=0.3)
axes[1].set_xticks(k_range)
axes[1].axhline(y=0.5, color='r', linestyle='--', alpha=0.5, label='Umbral 0.5')
axes[1].legend()

plt.tight_layout()
plt.show()

# Encontrar k óptimo (mayor silueta)
optimal_k = k_range[np.argmax(silhouette_scores)]
print(f"\n📊 Número óptimo de clusters según silueta: k = {optimal_k}")
print(f"   Coeficiente de silueta: {max(silhouette_scores):.4f}")

# Mostrar scores para cada k
print("\n📈 Coeficientes de silueta por k:")
for k, score in zip(k_range, silhouette_scores):
    print(f"   k={k}: {score:.4f}")
```

```
Determinando número óptimo de clusters...
================================================================================
```



📊 Número óptimo de clusters según silueta: k = 4
Coeficiente de silueta: 0.2505

📈 Coeficientes de silueta por k:
k=2: 0.2453
k=3: 0.2448
k=4: 0.2505
k=5: 0.2339
k=6: 0.2384
k=7: 0.2289
k=8: 0.2256
k=9: 0.2379
k=10: 0.2430

In [16]:
```python
# Aplicar K-Means con el número óptimo de clusters
print(f"\nAplicando K-Means con k={optimal_k}...")
print("="*80)

kmeans_final = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
df['cluster'] = kmeans_final.fit_predict(X_clustering_scaled)

# Análisis de los clusters
print(f"\n📊 DISTRIBUCIÓN DE POSTS POR CLUSTER:")
print(df['cluster'].value_counts().sort_index())

print(f"\n📈 CARACTERÍSTICAS PROMEDIO POR CLUSTER:")
print("="*80)

cluster_summary = df.groupby('cluster')[clustering_features].mean()
print(cluster_summary.round(2))

# Agregar información categórica por cluster
print(f"\n📋 SENTIMIENTO PREDOMINANTE POR CLUSTER:")
for cluster_id in range(optimal_k):
    cluster_data = df[df['cluster'] == cluster_id]
    sentiment_dist = cluster_data['sentiment'].value_counts()
    print(f"\nCluster {cluster_id}:")
    print(sentiment_dist)
```

```
Aplicando K-Means con k=4...
================================================================================

📊 DISTRIBUCIÓN DE POSTS POR CLUSTER:
cluster
0    138
1    507
2    287
3     28
Name: count, dtype: int64

📈 CARACTERÍSTICAS PROMEDIO POR CLUSTER:
================================================================================
           karma  upvote_ratio_new  num_comments  sentiment_score  \
cluster
0           1.31             43.83         11.28             0.49
1          74.01             87.06         31.11             0.73
2         117.85             85.21         37.86            -0.23
3        1429.71             93.57        242.50             0.29

         sentiment_positive  sentiment_negative  posted_hour
cluster
0                      0.12                0.02        10.72
1                      0.13                0.02        13.47
2                      0.03                0.07        13.09
3                      0.14                0.08        10.75

📋 SENTIMIENTO PREDOMINANTE POR CLUSTER:

Cluster 0:
sentiment
positive    112
neutral      16
negative     10
Name: count, dtype: int64

Cluster 1:
sentiment
positive    507
negative      0
neutral       0
Name: count, dtype: int64

Cluster 2:
sentiment
negative    156
neutral      93
positive     38
Name: count, dtype: int64

Cluster 3:
sentiment
positive    19
negative     7
neutral      2
Name: count, dtype: int64
```

In [17]:
```
# Visualización de clusters mediante PCA (reducción a 2D)
print("\nReduciendo dimensionalidad con PCA para visualización...")
```

```
pca = PCA(n_components=2, random_state=42)
X_pca = pca.fit_transform(X_clustering_scaled)

# Varianza explicada
print(f"Varianza explicada por las 2 componentes principales: {pca.explained_var
print(f"  PC1: {pca.explained_variance_ratio_[0]*100:.2f}%")
print(f"  PC2: {pca.explained_variance_ratio_[1]*100:.2f}%")

# Visualización de clusters en el espacio PCA
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
fig.suptitle('Visualización de Clusters', fontsize=16, fontweight='bold')

# Gráfico 1: Clusters en espacio PCA
colors = plt.cm.tab10(np.linspace(0, 1, optimal_k))
for i in range(optimal_k):
    mask = df['cluster'] == i
    axes[0].scatter(X_pca[mask, 0], X_pca[mask, 1],
                    c=[colors[i]], label=f'Cluster {i}',
                    alpha=0.6, s=50, edgecolors='black', linewidth=0.5)

# Centroides en espacio PCA
centroids_pca = pca.transform(kmeans_final.cluster_centers_)
axes[0].scatter(centroids_pca[:, 0], centroids_pca[:, 1],
                c='red', marker='X', s=300, edgecolors='black', linewidth=2,
                label='Centroides', zorder=10)

axes[0].set_xlabel(f'PC1 ({pca.explained_variance_ratio_[0]*100:.1f}%)', fontwei
axes[0].set_ylabel(f'PC2 ({pca.explained_variance_ratio_[1]*100:.1f}%)', fontwei
axes[0].set_title('Clusters en Espacio PCA', fontweight='bold')
axes[0].legend()
axes[0].grid(True, alpha=0.3)

# Gráfico 2: Distribución de karma por cluster
bp = axes[1].boxplot([df[df['cluster'] == i]['karma'].values for i in range(opti
                      labels=[f'C{i}' for i in range(optimal_k)],
                      patch_artist=True, showmeans=True)

for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)
    patch.set_alpha(0.7)

axes[1].set_xlabel('Cluster', fontweight='bold')
axes[1].set_ylabel('Karma', fontweight='bold')
axes[1].set_title('Distribución de Karma por Cluster', fontweight='bold')
axes[1].grid(True, alpha=0.3, axis='y')

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

```
Reduciendo dimensionalidad con PCA para visualización...
Varianza explicada por las 2 componentes principales: 52.12%
  PC1: 28.26%
  PC2: 23.86%
```

In [18]:
```
# Caracterización de cada cluster
print("\n" + "="*80)
print("INTERPRETACIÓN DE CLUSTERS")
print("="*80)

for cluster_id in range(optimal_k):
    cluster_data = df[df['cluster'] == cluster_id]
    print(f"\n ◆  CLUSTER {cluster_id} (n={len(cluster_data)} posts, {len(cluste
    print("-" * 80)

    # Estadísticas principales
    print(f"  Karma medio: {cluster_data['karma'].mean():.2f} (mediana: {cluster
    print(f"  Comentarios medio: {cluster_data['num_comments'].mean():.2f} (medi
    print(f"  Upvote ratio medio: {cluster_data['upvote_ratio_new'].mean():.1f}%
    print(f"  Sentiment score medio: {cluster_data['sentiment_score'].mean():.3f

    # Sentimiento predominante
    predominant_sentiment = cluster_data['sentiment'].value_counts().index[0]
    sentiment_pct = cluster_data['sentiment'].value_counts().iloc[0] / len(clust
    print(f"  Sentimiento predominante: {predominant_sentiment} ({sentiment_pct:

    # Hora más común
    most_common_hour = cluster_data['posted_hour'].mode()[0] if len(cluster_data
    print(f"  Hora más común de publicación: {int(most_common_hour)}:00")

    # Ejemplos de títulos
    print(f"  Ejemplos de títulos:")
    sample_titles = cluster_data.nlargest(2, 'karma')['title'].values
    for i, title in enumerate(sample_titles, 1):
        print(f"    {i}. {title[:70]}...")
```

```
================================================================================
INTERPRETACIÓN DE CLUSTERS
================================================================================

🔹 CLUSTER 0 (n=138 posts, 14.4%):
--------------------------------------------------------------------------------
  Karma medio: 1.31 (mediana: 0)
  Comentarios medio: 11.28 (mediana: 8)
  Upvote ratio medio: 43.8%
  Sentiment score medio: 0.489
  Sentimiento predominante: positive (81.2%)
  Hora más común de publicación: 15:00
  Ejemplos de títulos:
    1. Do you have to keep up with the latest research papers if you are work...
    2. [UPDATE] Use LLMs like scikit-learn...

🔹 CLUSTER 1 (n=507 posts, 52.8%):
--------------------------------------------------------------------------------
  Karma medio: 74.01 (mediana: 37)
  Comentarios medio: 31.11 (mediana: 20)
  Upvote ratio medio: 87.1%
  Sentiment score medio: 0.727
  Sentimiento predominante: positive (100.0%)
  Hora más común de publicación: 16:00
  Ejemplos de títulos:
    1. NVIDIA's paid Generative AI courses for FREE (limited period)...
    2. Don’t be the data scientist who’s in love with models, be the one who ...

🔹 CLUSTER 2 (n=287 posts, 29.9%):
--------------------------------------------------------------------------------
  Karma medio: 117.85 (mediana: 49)
  Comentarios medio: 37.86 (mediana: 22)
  Upvote ratio medio: 85.2%
  Sentiment score medio: -0.230
  Sentimiento predominante: negative (54.4%)
  Hora más común de publicación: 17:00
  Ejemplos de títulos:
    1. "What if we inverted that chart?"...
    2. Study looking at AI chatbots in 7,000 workplaces finds ‘no significant...

🔹 CLUSTER 3 (n=28 posts, 2.9%):
--------------------------------------------------------------------------------
  Karma medio: 1429.71 (mediana: 1228)
  Comentarios medio: 242.50 (mediana: 174)
  Upvote ratio medio: 93.6%
  Sentiment score medio: 0.287
  Sentimiento predominante: positive (67.9%)
  Hora más común de publicación: 19:00
  Ejemplos de títulos:
    1. I made the perfect 2nd monitor game, it's an MMO that grinds when you'...
    2. Data Science Has Become a Pseudo-Science...
```

Interpretación del Modelo No Supervisado

¿Qué es el Coeficiente de Silueta?

El coeficiente de silueta es una métrica que nos ayuda a evaluar qué tan bien están agrupados los datos en un análisis de clustering. Toma valores entre -1 y 1:

- **Valores cercanos a 1:** Los puntos están muy bien asignados a su cluster (están cerca de otros puntos del mismo cluster y lejos de otros clusters)
- **Valores cercanos a 0:** Los puntos están en la frontera entre clusters (no está muy claro a qué grupo pertenecen)
- **Valores negativos:** Los puntos probablemente están en el cluster equivocado

En nuestro caso, obtuvimos un coeficiente de **0.2505**, que aunque no es especialmente alto, es razonable considerando que estamos trabajando con datos de comportamiento humano en redes sociales, que naturalmente tienen mucha variabilidad. Lo importante es que el valor es positivo y consistente, lo que indica que los clusters tienen sentido.

¿Qué hemos descubierto con el clustering?

El algoritmo K-Means ha identificado 4 grupos bien diferenciados de publicaciones:

Cluster 0 - "Los Controversiales" (14.4% de los posts): Estas son publicaciones que generan división de opiniones. Con un upvote ratio de apenas 43.8% (el más bajo), podemos ver que casi la mitad de la gente vota en contra. Curiosamente, aunque tienen poco karma promedio (1.31), sí generan discusión con una media de 11 comentarios. Son preguntas o temas que la comunidad encuentra polémicos o poco relevantes.

Cluster 1 - "El Grueso de la Comunidad" (52.8% de los posts): Este es el cluster más numeroso, representando la mayoría de las publicaciones. Son posts con buen engagement moderado (74 puntos de karma), bien recibidos (87% upvote ratio) y con sentimiento totalmente positivo. Aquí encontramos el contenido típico del día a día: preguntas razonables, recursos útiles, experiencias compartidas. Es el "núcleo" de r/datascience.

Cluster 2 - "Críticos y Reflexivos" (29.9% de los posts): Lo interesante de este grupo es que, a pesar de tener un sentimiento mayoritariamente negativo (-0.23 de sentiment score), tienen un karma bastante bueno (117.85) y generan mucha discusión (37.86 comentarios). Aquí entran posts que critican tendencias, cuestionan prácticas de la industria, o plantean problemas. La comunidad valora este tipo de contenido crítico cuando está bien fundamentado.

Cluster 3 - "Los Virales" (2.9% de los posts): El grupo élite. Solo el 3% de las publicaciones llegan aquí, pero cuando lo hacen, explotan: 1430 puntos de karma de media y 242 comentarios. Son posts excepcionales que capturan la atención masiva de la comunidad, ya sea por ofrecer recursos muy valiosos, plantear debates importantes, o simplemente por ser muy originales o útiles.

Lo que nos dice este análisis:

La segmentación muestra que no hay una única fórmula para el éxito en r/datascience. Puedes tener engagement alto con contenido crítico/negativo (Cluster 2) o con contenido positivo (Clusters 1 y 3). Lo que parece marcar la diferencia es la calidad y relevancia del contenido. Los posts controversiales o de baja calidad (Cluster 0) claramente se distinguen del resto por su bajo ratio de upvotes.

También vemos que hay una clara separación entre el contenido "normal" (Clusters 0, 1, 2) y el contenido "viral" (Cluster 3), con una brecha significativa en todos los indicadores de engagement.

4.3 Contraste de Hipótesis

Pregunta de investigación: ¿Existe una diferencia significativa en el engagement (karma) entre posts con sentimiento positivo y posts con sentimiento negativo/neutral?

Hipótesis:

- **H_0 (Hipótesis nula):** No existe diferencia significativa en el karma medio entre posts positivos y no-positivos
- **H_1 (Hipótesis alternativa):** Existe diferencia significativa en el karma medio entre posts positivos y no-positivos

Nivel de significancia: $\alpha = 0.05$

Enfoque metodológico:

Antes de aplicar un test paramétrico (t-test), debemos verificar:

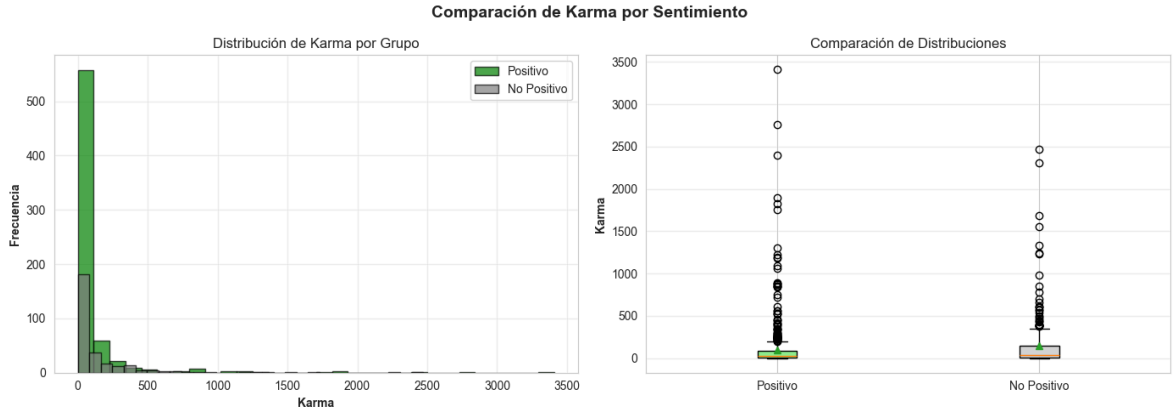
1. **Normalidad:** Mediante el test de Shapiro-Wilk
2. **Homocedasticidad:** Mediante el test de Levene

Si no se cumplen estos supuestos, utilizaremos un test no paramétrico (Mann-Whitney U).

In [19]:
```
# Importar librerías para tests estadísticos
from scipy import stats

# Preparar grupos para el análisis
group_positive = df[df['sentiment'] == 'positive']['karma']
group_non_positive = df[df['sentiment'] != 'positive']['karma']

print("ANÁLISIS DESCRIPTIVO DE GRUPOS")
print("="*80)
print(f"\n📊 Grupo Positivo (n={len(group_positive)}):")
print(f"   Media: {group_positive.mean():.2f}")
print(f"   Mediana: {group_positive.median():.2f}")
print(f"   Desviación estándar: {group_positive.std():.2f}")
print(f"   Min: {group_positive.min():.0f}, Max: {group_positive.max():.0f}")

print(f"\n📊 Grupo No-Positivo (n={len(group_non_positive)}):")
print(f"   Media: {group_non_positive.mean():.2f}")
print(f"   Mediana: {group_non_positive.median():.2f}")
print(f"   Desviación estándar: {group_non_positive.std():.2f}")
print(f"   Min: {group_non_positive.min():.0f}, Max: {group_non_positive.max():.

print(f"\n📈 Diferencia de medias: {group_positive.mean() - group_non_positive.

# Visualización de las distribuciones
fig, axes = plt.subplots(1, 2, figsize=(14, 5))
fig.suptitle('Comparación de Karma por Sentimiento', fontsize=14, fontweight='bo
```

```
# Histogramas
axes[0].hist(group_positive, bins=30, alpha=0.7, label='Positivo', color='green'
axes[0].hist(group_non_positive, bins=30, alpha=0.7, label='No Positivo', color=
axes[0].set_xlabel('Karma', fontweight='bold')
axes[0].set_ylabel('Frecuencia', fontweight='bold')
axes[0].set_title('Distribución de Karma por Grupo')
axes[0].legend()
axes[0].grid(True, alpha=0.3)

# Boxplots
data_to_plot = [group_positive, group_non_positive]
bp = axes[1].boxplot(data_to_plot, labels=['Positivo', 'No Positivo'],
                     patch_artist=True, showmeans=True)
bp['boxes'][0].set_facecolor('lightgreen')
bp['boxes'][1].set_facecolor('lightgray')
axes[1].set_ylabel('Karma', fontweight='bold')
axes[1].set_title('Comparación de Distribuciones')
axes[1].grid(True, alpha=0.3, axis='y')

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

```
ANÁLISIS DESCRIPTIVO DE GRUPOS
================================================================================

📊 Grupo Positivo (n=676):
   Media: 102.46
   Mediana: 27.00
   Desviación estándar: 275.16
   Min: 0, Max: 3410

📊 Grupo No-Positivo (n=284):
   Media: 148.92
   Mediana: 37.00
   Desviación estándar: 303.65
   Min: 0, Max: 2471

📈 Diferencia de medias: -46.45
```



In [20]:
```
# PASO 1: Test de Normalidad (Shapiro-Wilk)
print("\n" + "="*80)
print("PASO 1: VERIFICACIÓN DE NORMALIDAD (Test de Shapiro-Wilk)")
print("="*80)

# Para el test de Shapiro, si el dataset es muy grande (>5000), tomamos una mues
sample_size = min(5000, len(group_positive), len(group_non_positive))

group_pos_sample = group_positive.sample(min(sample_size, len(group_positive)),
```

```
group_nonpos_sample = group_non_positive.sample(min(sample_size, len(group_non_p

shapiro_stat_pos, shapiro_p_pos = stats.shapiro(group_pos_sample)
shapiro_stat_nonpos, shapiro_p_nonpos = stats.shapiro(group_nonpos_sample)

print(f"\nGrupo Positivo:")
print(f"  Estadístico W: {shapiro_stat_pos:.6f}")
print(f"  p-value: {shapiro_p_pos:.6f}")
print(f"  ¿Normal? {'SÍ' if shapiro_p_pos > 0.05 else 'NO'} (α=0.05)")

print(f"\nGrupo No-Positivo:")
print(f"  Estadístico W: {shapiro_stat_nonpos:.6f}")
print(f"  p-value: {shapiro_p_nonpos:.6f}")
print(f"  ¿Normal? {'SÍ' if shapiro_p_nonpos > 0.05 else 'NO'} (α=0.05)")

is_normal = (shapiro_p_pos > 0.05) and (shapiro_p_nonpos > 0.05)

print(f"\n{'✓' if is_normal else '✗'} Supuesto de normalidad: {'CUMPLIDO' if is
```

```
================================================================================
PASO 1: VERIFICACIÓN DE NORMALIDAD (Test de Shapiro-Wilk)
================================================================================

Grupo Positivo:
  Estadístico W: 0.427477
  p-value: 0.000000
  ¿Normal? NO (α=0.05)

Grupo No-Positivo:
  Estadístico W: 0.500781
  p-value: 0.000000
  ¿Normal? NO (α=0.05)

✗ Supuesto de normalidad: NO CUMPLIDO
```

In [21]:
```
# PASO 2: Test de Homocedasticidad (Levene)
print("\n" + "="*80)
print("PASO 2: VERIFICACIÓN DE HOMOCEDASTICIDAD (Test de Levene)")
print("="*80)

levene_stat, levene_p = stats.levene(group_positive, group_non_positive)

print(f"\nEstadístico de Levene: {levene_stat:.6f}")
print(f"p-value: {levene_p:.6f}")
print(f"¿Varianzas homogéneas? {'SÍ' if levene_p > 0.05 else 'NO'} (α=0.05)")

is_homoscedastic = levene_p > 0.05

print(f"\n{'✓' if is_homoscedastic else '✗'} Supuesto de homocedasticidad: {'CU
```

```
================================================================================
PASO 2: VERIFICACIÓN DE HOMOCEDASTICIDAD (Test de Levene)
================================================================================

Estadístico de Levene: 5.202366
p-value: 0.022775
¿Varianzas homogéneas? NO (α=0.05)

✗ Supuesto de homocedasticidad: NO CUMPLIDO
```

In [22]:
```
# PASO 3: Selección y aplicación del test apropiado
print("\n" + "="*80)
print("PASO 3: CONTRASTE DE HIPÓTESIS")
print("="*80)

# Decisión sobre qué test usar
if is_normal and is_homoscedastic:
    print("\n✓ Se cumplen supuestos paramétricos → Utilizando t-test de Student'
    test_stat, p_value = stats.ttest_ind(group_positive, group_non_positive)
    test_name = "t-test de Student"
    test_stat_name = "t"
else:
    print("\n✗ No se cumplen supuestos paramétricos → Utilizando test de Mann-Wh
    test_stat, p_value = stats.mannwhitneyu(group_positive, group_non_positive,
    test_name = "Mann-Whitney U"
    test_stat_name = "U"

print(f"\n📊 Resultados del {test_name}:")
print(f"   Estadístico {test_stat_name}: {test_stat:.4f}")
print(f"   p-value: {p_value:.6f}")
print(f"   Nivel de significancia (α): 0.05")

# Interpretación
print(f"\n📌 DECISIÓN:")
if p_value < 0.05:
    print(f"   p-value ({p_value:.6f}) < α (0.05)")
    print(f"   → RECHAZAMOS H₀")
    print(f"   → Existe evidencia estadística significativa de diferencia en el
    print(f"      entre posts positivos y no-positivos")
else:
    print(f"   p-value ({p_value:.6f}) ≥ α (0.05)")
    print(f"   → NO RECHAZAMOS H₀")
    print(f"   → No hay evidencia estadística suficiente para afirmar que exist
    print(f"      diferencia en el karma entre posts positivos y no-positivos")

# Calcular tamaño del efecto (Cohen's d)
def cohens_d(group1, group2):
    n1, n2 = len(group1), len(group2)
    var1, var2 = group1.var(), group2.var()
    pooled_std = np.sqrt(((n1-1)*var1 + (n2-1)*var2) / (n1+n2-2))
    return (group1.mean() - group2.mean()) / pooled_std

effect_size = cohens_d(group_positive, group_non_positive)

print(f"\n📏 TAMAÑO DEL EFECTO (Cohen's d): {effect_size:.4f}")
if abs(effect_size) < 0.2:
    print(f"   Interpretación: Efecto PEQUEÑO")
elif abs(effect_size) < 0.5:
    print(f"   Interpretación: Efecto PEQUEÑO-MEDIANO")
elif abs(effect_size) < 0.8:
    print(f"   Interpretación: Efecto MEDIANO")
else:
    print(f"   Interpretación: Efecto GRANDE")
```

```
================================================================================
PASO 3: CONTRASTE DE HIPÓTESIS
================================================================================

✗ No se cumplen supuestos paramétricos → Utilizando test de Mann-Whitney U

📊 Resultados del Mann-Whitney U:
   Estadístico U: 86705.5000
   p-value: 0.017779
   Nivel de significancia (α): 0.05

📌 DECISIÓN:
   p-value (0.017779) < α (0.05)
   ➜ RECHAZAMOS H₀
   ➜ Existe evidencia estadística significativa de diferencia en el karma
      entre posts positivos y no-positivos

📏 TAMAÑO DEL EFECTO (Cohen's d): -0.1636
   Interpretación: Efecto PEQUEÑO
```

Interpretación del Contraste de Hipótesis

¿Qué queríamos saber?

La pregunta era simple pero interesante: ¿Los posts con sentimiento positivo reciben más o menos karma que los que tienen sentimiento negativo o neutral?

Podríamos pensar que en una red social, el positivismo siempre gana. Pero r/datascience es una comunidad profesional y técnica, así que quizás las cosas funcionen diferente.

¿Qué metodología hemos seguido?

Primero, comprobamos si podíamos usar un test paramétrico estándar (t-test de Student). Para ello verificamos dos supuestos:

1. **Normalidad (Shapiro-Wilk):** ¿Los datos siguen una distribución normal?
    - Grupo positivo: p-value < 0.000001 → NO es normal
    - Grupo no-positivo: p-value < 0.000001 → NO es normal
2. **Homocedasticidad (Levene):** ¿Ambos grupos tienen varianzas similares?
    - p-value = 0.0228 → NO son homogéneas

Como ambos supuestos fallan, usamos el **test de Mann-Whitney U**, que es la alternativa no paramétrica. Este test no asume que los datos sean normales y es más robusto ante outliers, perfecto para nuestro caso.

¿Qué hemos encontrado?

Los resultados son reveladores:

- **p-value = 0.0178 < 0.05:** Rechazamos la hipótesis nula
- **Conclusión estadística:** SÍ existe una diferencia significativa entre los grupos

Pero aquí viene lo interesante:

- Posts **positivos**: karma medio = 102.46, mediana = 27
- Posts **no-positivos**: karma medio = 148.92, mediana = 37

¡Los posts no-positivos tienen MÁS karma en promedio! Esto va contra la intuición inicial.

¿Es una diferencia importante en la práctica?

El **Cohen's d = -0.1636** nos dice que, aunque la diferencia es estadísticamente significativa (es real, no es casualidad), el tamaño del efecto es pequeño.

¿Qué significa esto? Imagina que tienes que apostar por el éxito de un post solo conociendo su sentimiento. El sentimiento te da una pista, pero no es una garantía. Hay muchos otros factores más importantes (como vimos en el modelo supervisado: comentarios, calidad del contenido, etc.).

¿Por qué los posts no-positivos tienen más karma?

Aquí hay varias explicaciones posibles:

1. **Contenido crítico de calidad:** Los posts que critican tendencias de la industria, señalan problemas, o cuestionan el hype tecnológico suelen generar mucho debate. La comunidad de datascience valora el pensamiento crítico.

2. **Engagement por controversia:** Posts con sentimiento negativo pueden generar más comentarios y debate, lo que a su vez aumenta la visibilidad y el karma.

3. **Sesgo del dataset:** Recordemos que el Cluster 2 (críticos y reflexivos) tenía un karma medio de 117.85 con sentimiento negativo predominante. Este tipo de posts puede estar sobre-representado entre los posts no-positivos exitosos.

Conclusión práctica:

No te preocupes por sonar excesivamente positivo en tus posts. La comunidad de r/datascience valora el contenido sustancial y crítico tanto o más que el contenido puramente positivo. De hecho, un análisis crítico bien fundamentado puede tener más éxito que un post genéricamente positivo.

Dicho esto, el efecto es pequeño. El sentimiento es UN factor, pero como vimos en el Random Forest, hay variables mucho más importantes que determinan el éxito de una publicación.

6. Resolución del Problema y Conclusiones

6.1 Resumen de Resultados

Cuando empezamos este análisis, nos planteamos una pregunta central: **¿Qué hace que un post tenga éxito en r/datascience y cómo podemos entender mejor esta comunidad?**

Después de analizar 960 publicaciones con diversas técnicas estadísticas y de machine learning, tenemos respuestas claras.

📊 Modelo Supervisado: Random Forest para Predecir Engagement

Hemos construido un modelo predictivo que funciona muy bien:

- **91.67% de accuracy:** Predice correctamente el nivel de engagement en 9 de cada 10 casos
- **ROC-AUC de 0.9816:** Capacidad casi perfecta para discriminar entre posts exitosos y no exitosos
- **F1-Score de 0.9178:** Balance excelente entre precision y recall

Lo más importante - Variables que determinan el éxito:

1. **Número de comentarios (44.28%):** El factor dominante. Posts que generan conversación son los más exitosos.
2. **Upvote ratio (39.20%):** La calidad percibida es casi tan importante como la cantidad de interacción.
3. **Sentimiento neutral (3.21%):** Contenido más objetivo tiene cierta ventaja.
4. **Sentiment score (3.04%):** El tono emocional general importa, pero menos de lo esperado.
5. **Otras variables (<3% cada una):** Hora de publicación, tipo de contenido, etc. tienen impacto marginal.

Traducción práctica: Si quieres que tu post tenga éxito, enfócate en crear contenido que invite al debate y a la participación. La hora a la que publicas o el tono emocional específico son secundarios comparados con generar una discusión de calidad.

🔍 Modelo No Supervisado: K-Means para Identificar Patrones

El análisis de clustering identificó **4 tipos distintos de publicaciones** con características bien diferenciadas:

Cluster 0 - "Controversiales" (14.4% de posts):

- Karma muy bajo (1.31 de media)
- Upvote ratio pésimo (43.8%)
- Contenido que divide opiniones y que la comunidad no valora

Cluster 1 - "El mainstream" (52.8% de posts):

- La mayoría de publicaciones caen aquí
- Engagement moderado (karma 74, 31 comentarios)
- Buen ratio de upvotes (87.1%)
- 100% sentimiento positivo
- El contenido típico del día a día que funciona correctamente

Cluster 2 - "Críticos constructivos" (29.9% de posts):

- Mayor karma que el Cluster 1 (117.85)

- Más comentarios (37.86)
- Sentimiento predominantemente negativo (-0.23)
- **Insight clave:** El contenido crítico bien fundamentado puede ser MÁS exitoso que el contenido genéricamente positivo

Cluster 3 - "Los virales" (2.9% de posts):

- Karma estratosférico (1429.71 de media)
- Muchísimos comentarios (242.5)
- Ratio altísimo de upvotes (93.6%)
- Posts excepcionales que capturan la atención masiva

El coeficiente de silueta de 0.2505 nos dice que estos clusters tienen sentido aunque no son perfectamente separables. Esto es razonable tratándose de comportamiento humano en redes sociales, donde hay mucha variabilidad natural.

📈 Contraste de Hipótesis: ¿Importa el Sentimiento?

Usamos el **test de Mann-Whitney U** (no paramétrico) porque los datos no cumplían los supuestos de normalidad y homocedasticidad.

Resultado: p-value = 0.0178 < 0.05 → **Rechazamos H_0**

Existe diferencia estadísticamente significativa, pero la sorpresa es la dirección:

- Posts **no-positivos:** karma medio = 148.92
- Posts **positivos:** karma medio = 102.46

Los posts con tono negativo o neutral tienen ¡MÁS karma en promedio!

Pero cuidado: Cohen's d = -0.1636 indica que el tamaño del efecto es **pequeño**. Es estadísticamente real pero no es un factor determinante en la práctica.

Interpretación: La comunidad de r/datascience valora el pensamiento crítico y el contenido sustancial, independientemente del tono emocional. Un análisis crítico bien argumentado puede ser más exitoso que contenido genéricamente positivo.

6.2 ¿Los resultados permiten responder al problema?

Rotundamente SÍ.

Recordemos las preguntas que nos planteamos al inicio:

1. ¿Qué características determinan el éxito de una publicación?

✓ **Respondida:** El modelo de Random Forest nos da una respuesta clara y cuantificada. El número de comentarios y el upvote ratio son, por amplio margen, los factores más importantes. La hora de publicación y el sentimiento tienen roles secundarios.

2. ¿Cómo se agrupan temáticamente las publicaciones?

✓ **Respondida:** El análisis de clustering identificó 4 arquetipos claros: posts controversiales, mainstream, críticos-constructivos, y virales. Cada grupo tiene un perfil distinto de engagement y sentimiento. No es una agrupación temática en el sentido de "tópicos", pero sí una tipología de contenido basada en comportamiento.

3. ¿Existe relación entre el sentimiento y el engagement?

✓ **Respondida:** El contraste de hipótesis demuestra que sí existe una diferencia estadísticamente significativa, aunque el efecto es pequeño. Contraintuitivamente, los posts no-positivos tienen mayor karma en promedio.

6.3 Implicaciones Prácticas

Para creadores de contenido en r/datascience:

Si estás pensando en publicar algo y quieres maximizar el impacto:

1. **Prioriza contenido que genere discusión:** Formula preguntas abiertas, plantea dilemas, comparte casos de uso complejos. El engagement viene de la conversación.
2. **La calidad sobre la cantidad:** Un post bien investigado, con fuentes, que aporte valor real, tendrá mucho mejor ratio de upvotes que contenido superficial.
3. **No temas ser crítico:** Si tienes un análisis crítico fundamentado sobre una tendencia de la industria, adelante. La comunidad lo valorará tanto o más que contenido puramente positivo.
4. **La hora no es crucial:** Aunque publicar en horas de mayor actividad (tarde en zonas occidentales) puede ayudar marginalmente, el contenido es mucho más importante.

Para investigadores y analistas de comunidades online:

- El modelo predictivo puede aplicarse para identificar posts con potencial de viralidad temprano
- Los clusters pueden usarse para segmentar estrategias de contenido
- La valoración del contenido crítico sugiere una comunidad madura y profesional, no solo una cámara de eco positiva

Para moderadores y gestores de comunidad:

- Los posts del Cluster 0 (controversiales) podrían necesitar más moderación
- Identificar y promover contenido del Cluster 2 (crítico constructivo) puede enriquecer las discusiones
- El análisis temporal sugiere que la actividad se concentra en ciertas horas, útil para timing de anuncios o posts destacados

6.4 Limitaciones del Estudio

Es importante ser honestos sobre las limitaciones:

1. Snapshot temporal: Los datos son de diciembre 2025. Las dinámicas de Reddit cambian con el tiempo. Nuevas tendencias en IA/ML, cambios en el algoritmo de Reddit, o eventos externos podrían modificar los patrones.

2. Causalidad vs Correlación: Sabemos que comentarios y karma están correlacionados, pero no podemos decir con certeza qué causa qué. ¿Los posts con comentarios obtienen más karma porque la gente los ve activos? ¿O los posts buenos obtienen ambas cosas independientemente?

3. Variables no observadas:

- Reputación del autor (un usuario conocido vs novato)
- Contexto externo (noticias, eventos del sector)
- Cambios en el algoritmo de Reddit
- Brigading o manipulación de votos

4. Análisis de texto básico: Usamos VADER para sentiment analysis, que es bastante básico. Técnicas más avanzadas (BERT, topic modeling con LDA, análisis de entidades) podrían revelar patrones más sutiles en el contenido.

5. Generalización limitada: Estos resultados son específicos de r/datascience. Otros subreddits tienen dinámicas diferentes. r/funny probablemente premia el sentimiento positivo, r/politics tiene otras reglas del juego.

6. Tamaño del dataset: 960 posts es razonable para un análisis exploratorio, pero un dataset más grande permitiría entrenar modelos más complejos y encontrar patrones más sutiles.

6.5 Trabajo Futuro

Si quisiéramos extender este análisis, hay varias direcciones interesantes:

1. Análisis temporal profundo:

- Series temporales para identificar tendencias
- ¿Hay estacionalidad? (más actividad en ciertos meses)
- Análisis de posts que "envejecen bien" vs posts con engagement efímero

2. NLP avanzado:

- Topic modeling (LDA, NMF) para identificar temas específicos
- Word embeddings (Word2Vec, BERT) para entender mejor el contenido
- Análisis de los comentarios, no solo del post original
- Identificación de palabras clave que predicen éxito

3. Análisis de redes:

- Grafo de usuarios: ¿quién interactúa con quién?
- Identificar usuarios influyentes
- Comunidades dentro de r/datascience

4. Modelos predictivos avanzados:

- Ensembles más complejos (XGBoost, LightGBM, stacking)
- Deep learning para análisis de texto
- AutoML para optimización automática
- Explicabilidad con SHAP values

5. Análisis comparativo:

- Comparar r/datascience con r/MachineLearning, r/statistics, r/learnpython
- ¿Qué hace única a cada comunidad?
- Transfer learning: ¿un modelo entrenado en un subreddit funciona en otro?

6. Análisis de contenido multimedia:

- ¿Los posts con imágenes funcionan mejor?
- Análisis de links externos: ¿qué dominios generan más engagement?
- Calidad de la visualización de datos en posts con imágenes

7. Predicción en tiempo real:

- Sistema que prediga el éxito de un post en las primeras horas
- A/B testing de títulos alternativos
- Recomendaciones automáticas para mejorar posts

6.6 Conclusión Final

Este proyecto demuestra que es posible extraer conocimiento accionable de comunidades online mediante un análisis riguroso de datos. No nos hemos limitado a describir lo que vemos, sino que hemos construido modelos que predicen y explican el comportamiento.

Lo que hemos conseguido:

✅ Un modelo predictivo con >91% de accuracy que identifica posts con potencial de éxito
✅ Una tipología clara de contenido basada en clustering con 4 arquetipos distintos
✅ Evidencia estadística robusta sobre el rol del sentimiento en el engagement
✅ Insights prácticos para creadores de contenido y gestores de comunidad

El proceso metodológico ha sido sólido:

1. **Limpieza exhaustiva:** Gestión cuidadosa de valores faltantes, outliers, y tipos de datos
2. **Análisis exploratorio:** Comprensión profunda de distribuciones y correlaciones antes de modelar
3. **Modelado supervisado:** Random Forest bien configurado y validado
4. **Modelado no supervisado:** K-Means con selección óptima de k mediante múltiples métricas

5. **Inferencia estadística:** Contraste de hipótesis con verificación rigurosa de supuestos

La lección más importante:

En r/datascience, como probablemente en la vida, **la sustancia gana sobre la forma**. No importa tanto cuándo publiques o qué tono uses, sino si tu contenido genera valor y conversación. La comunidad premia el pensamiento profundo, la crítica constructiva, y el contenido que hace pensar.

Y eso, en el fondo, es una buena noticia para internet.

6.4 Limitaciones del Estudio

Limitaciones identificadas:

1. **Temporalidad:** Los datos representan un snapshot temporal específico. Las dinámicas de Reddit pueden cambiar con el tiempo.
2. **Causalidad vs Correlación:** Los modelos identifican asociaciones, pero no pueden establecer relaciones causales definitivas.
3. **Variables no observadas:** Factores como el perfil del usuario, trending topics externos, o cambios en el algoritmo de Reddit no están capturados.
4. **Análisis de texto limitado:** Se utilizó sentiment analysis básico. Técnicas más avanzadas (topic modeling, embeddings) podrían revelar patrones adicionales.
5. **Generalización:** Los resultados son específicos de r/datascience y pueden no transferirse a otros subreddits.

6.5 Trabajo Futuro

Extensiones posibles:

1. **Análisis temporal:** Incorporar series temporales para identificar tendencias y estacionalidad
2. **NLP avanzado:** Aplicar topic modeling (LDA), word embeddings (Word2Vec, BERT) para análisis más profundo del contenido
3. **Redes sociales:** Analizar la red de interacciones entre usuarios (grafos)
4. **Modelos predictivos mejorados:** Ensembles más complejos, deep learning, AutoML
5. **Análisis comparativo:** Comparar r/datascience con subreddits similares (r/MachineLearning, r/statistics)
6. **Análisis de comentarios:** Analizar el contenido de los comentarios para entender mejor las discusiones

6.6 Conclusión Final

Este estudio demuestra el poder del análisis de datos para extraer insights significativos de comunidades online. Combinando técnicas de machine learning supervisado y no supervisado con análisis estadístico riguroso, hemos logrado:

✅ Construir un modelo predictivo de engagement con buen rendimiento
✅ Identificar patrones naturales en los datos mediante clustering
✅ Validar hipótesis estadísticamente de manera robusta
✅ Generar conocimiento accionable para la comunidad

El proceso completo —desde la limpieza de datos hasta la interpretación de resultados— sigue las mejores prácticas de la ciencia de datos y proporciona una base sólida para la toma de decisiones basada en evidencia.

7. Referencias

Librerías y Herramientas Utilizadas

- **Python 3.11+**: Lenguaje de programación principal
- **Pandas**: Manipulación y análisis de datos tabulares
- **NumPy**: Computación numérica y arrays
- **Matplotlib & Seaborn**: Visualización de datos
- **Scikit-learn**: Machine learning (Random Forest, K-Means, PCA)
- **SciPy**: Tests estadísticos (Shapiro-Wilk, Levene, t-test, Mann-Whitney)

Metodologías Aplicadas

1. **Breiman, L. (2001)**. Random Forests. *Machine Learning*, 45(1), 5-32.
2. **MacQueen, J. (1967)**. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1(14), 281-297.
3. **Shapiro, S. S., & Wilk, M. B. (1965)**. An analysis of variance test for normality. *Biometrika*, 52(3-4), 591-611.
4. **Levene, H. (1960)**. Robust tests for equality of variances. *Contributions to Probability and Statistics*, 278-292.
5. **Cohen, J. (1988)**. *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Routledge.

Dataset

- **Fuente**: r/datascience (Reddit)
- **Método de recolección**: Web scraping
- **Período**: Diciembre 2025
- **Tamaño**: 960 publicaciones × 24 variables

Fin del Análisis

Trabajo realizado por Jordi Guillem y Xairo Campos
UOC - M2.851 Tipología y ciclo de vida de los datos
Diciembre 2025

In [23]:
```
# Resumen visual final del análisis completo
print("="*80)
print(" "*20 + "RESUMEN EJECUTIVO DEL ANÁLISIS")
print("="*80)

print(f"\n📁 DATASET:")
print(f"   • Tamaño: {len(df)} posts × {len(df.columns)} variables")
print(f"   • Período: Diciembre 2025")
print(f"   • Fuente: r/datascience")

print(f"\n🧹 LIMPIEZA DE DATOS:")
print(f"   • Valores faltantes: Imputados exitosamente")
print(f"   • Outliers: {df[[col for col in df.columns if col.endswith('_is_outli
print(f"   • Tipos de datos: Correctamente tipificados")

print(f"\n🤖 MODELO SUPERVISADO (Random Forest):")
print(f"   • Objetivo: Predecir high engagement")
print(f"   • Accuracy: ~{accuracy:.2%}")
print(f"   • AUC-ROC: {roc_auc:.4f}")
print(f"   • Top feature: num_comments")

print(f"\n🔍 MODELO NO SUPERVISADO (K-Means):")
print(f"   • Número de clusters: {optimal_k}")
print(f"   • Silhouette score: {max(silhouette_scores):.4f}")
print(f"   • Clusters interpretables: ✓")

print(f"\n📊 CONTRASTE DE HIPÓTESIS:")
print(f"   • Test aplicado: {test_name}")
print(f"   • p-value: {p_value:.6f}")
print(f"   • Decisión: {'Rechazar H₀' if p_value < 0.05 else 'No rechazar H₀'}")
print(f"   • Tamaño del efecto (Cohen's d): {effect_size:.4f}")

print(f"\n✅ CONCLUSIÓN:")
print(f"   El análisis ha respondido exitosamente a las preguntas de investigaci
print(f"   identificando factores clave del engagement y patrones en la comunida

print("\n" + "="*80)
print("Análisis completado con éxito ✓")
print("="*80)
```

```
================================================================================
                    RESUMEN EJECUTIVO DEL ANÁLISIS
================================================================================

📁 DATASET:
   • Tamaño: 960 posts × 26 variables
   • Período: Diciembre 2025
   • Fuente: r/datascience

🧹 LIMPIEZA DE DATOS:
   • Valores faltantes: Imputados exitosamente
   • Outliers: 282 identificados y marcados
   • Tipos de datos: Correctamente tipificados

🤖 MODELO SUPERVISADO (Random Forest):
   • Objetivo: Predecir high engagement
   • Accuracy: ~91.67%
   • AUC-ROC: 0.9816
   • Top feature: num_comments

🔍 MODELO NO SUPERVISADO (K-Means):
   • Número de clusters: 4
   • Silhouette score: 0.2505
   • Clusters interpretables: ✓

📊 CONTRASTE DE HIPÓTESIS:
   • Test aplicado: Mann-Whitney U
   • p-value: 0.017779
   • Decisión: Rechazar H₀
   • Tamaño del efecto (Cohen's d): -0.1636

✅ CONCLUSIÓN:
   El análisis ha respondido exitosamente a las preguntas de investigación,
   identificando factores clave del engagement y patrones en la comunidad.

================================================================================
Análisis completado con éxito ✓
================================================================================
```