Módulo 3: Análisis Avanzado y Modelado Predictivo para Indicadores

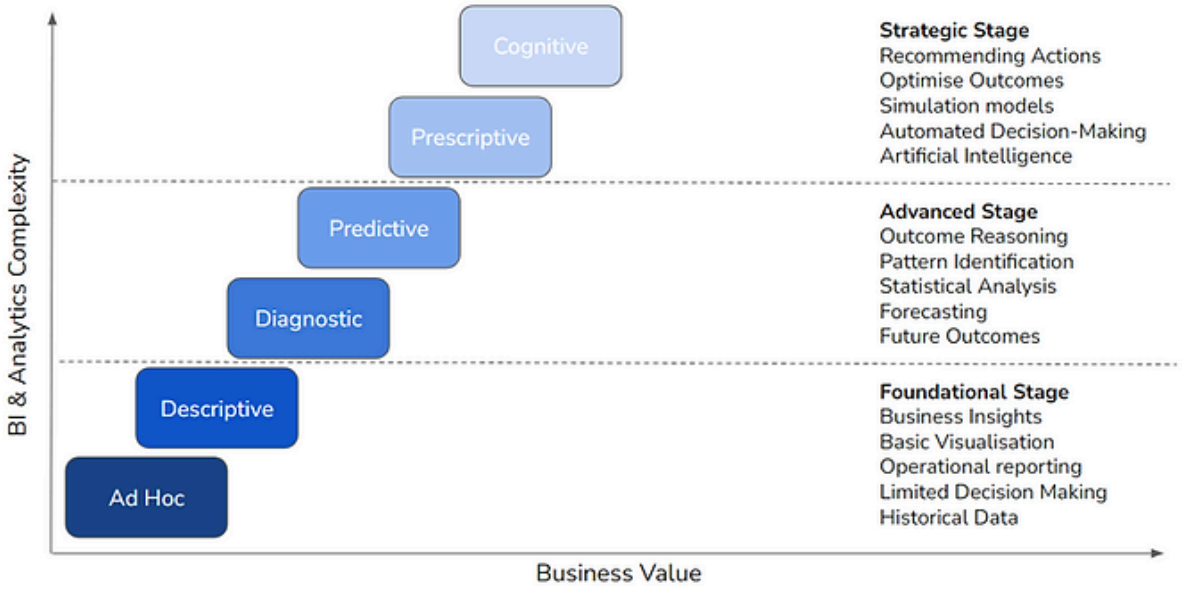
Este módulo se centra en profundizar en técnicas de análisis avanzadas y aplicaciones predictivas específicas para KPIs en contextos empresariales. Aprenderás sobre el análisis descriptivo multivariable, regresiones multivariables y análisis predictivo utilizando series de tiempo.

Introducción a la Madurez Analítica en las Empresas

La madurez analítica de una empresa se refiere al grado de habilidad, competencia y tecnología que una organización ha alcanzado en el campo del análisis de datos y business intelligence. Las empresas pueden clasificarse en diferentes niveles de madurez analítica, desde aquellas con capacidades básicas de informes descriptivos hasta aquellas que utilizan análisis avanzados para influir en las decisiones estratégicas y operativas de manera proactiva.

Niveles de Madurez Analítica (Según Gartner):

El concepto de madurez analítica en empresas se refiere a la capacidad de una organización para utilizar datos analíticos de manera efectiva para informar decisiones estratégicas. La madurez analítica se puede dividir en varios niveles, cada uno reflejando una mayor capacidad y sofisticación en el uso de datos y análisis.



Nivel 0) Empresas con Madurez Analítica Nula

- Características: Estas empresas carecen de prácticas estructuradas de recopilación o análisis de datos. Sus decisiones se basan en la intuición o en experiencias pasadas sin apoyo empírico.

- Acciones Recomendadas:
  - Establecer la recopilación de datos: Iniciar la implementación de sistemas para la recopilación sistemática de datos operativos y de clientes.
  - Capacitación básica en datos: Invertir en formación básica para los empleados en la interpretación y uso de datos en la toma de decisiones.

Nivel 1) Analítica Descriptiva

- Descripción: Las empresas en este nivel utilizan datos para describir lo que ha sucedido, generando informes estáticos o dashboards que resumen actividades pasadas.
- Ejemplo: Un retailer podría utilizar la analítica descriptiva para reportar las ventas totales del último trimestre.
- Acciones Recomendadas:
  - Automatizar informes: Desarrollar dashboards automatizados que proporcionen visibilidad constante sobre indicadores clave de rendimiento.
  - Formación en herramientas analíticas: Proporcionar acceso a herramientas de BI (Business Intelligence) como Tableau o Power BI para facilitar la visualización de datos.

Nivel 2) Analítica Diagnóstica

- Descripción: Este nivel se centra en el análisis de por qué ocurrieron ciertos eventos, utilizando técnicas como la minería de datos para identificar patrones y correlaciones.
- Ejemplo: Una empresa de software podría analizar las razones de la deserción de clientes analizando patrones de uso del producto y feedback de servicio al cliente.
- Acciones Recomendadas:
  - Implementar herramientas de análisis estadístico: Emplear software que permita realizar pruebas de hipótesis y análisis de correlación.
  - Capacitación en análisis causal: Educar a los empleados sobre métodos estadísticos que pueden aplicar para identificar causas subyacentes de los problemas detectados.

Nivel 3) Analítica Predictiva

- Descripción: Las empresas utilizan modelos estadísticos y de machine learning para predecir futuros eventos basados en datos históricos.
- Ejemplo: Un banco puede utilizar modelos predictivos para identificar clientes con alto riesgo de impago de créditos.
- Acciones Recomendadas:
  - Desarrollar capacidades de modelado: Construir equipos internos o asociarse con terceros para desarrollar competencias en modelado estadístico y machine learning.
  - Incorporar la experimentación: Fomentar una cultura de prueba y error donde se utilicen modelos predictivos para testear diferentes estrategias de negocio.

Nivel 4) Analítica Prescriptiva

- Descripción: Este es el nivel más avanzado, donde no solo se predice lo que sucederá, sino que también se prescriben acciones específicas.

- Ejemplo: Una empresa de e-commerce podría utilizar la analítica prescriptiva para optimizar automáticamente los precios de los productos en tiempo real, basándose en la demanda y el comportamiento de compra.
- Acciones Recomendadas:
  - Implementar sistemas de decisión automatizados: Utilizar algoritmos avanzados que no solo predigan resultados, sino que también tomen decisiones en tiempo real.
  - Integración profunda de IA: Integrar soluciones de inteligencia artificial que puedan aprender y adaptarse continuamente a nuevas condiciones y datos para mejorar la toma de decisiones.

Estos niveles de madurez no solo representan un mayor uso de tecnología y datos, sino también un cambio cultural hacia una organización más orientada a datos y respaldada por decisiones empíricas. Empresas como Gartner y McKinsey proporcionan frameworks y estudios detallados que pueden servir como guía para las organizaciones que buscan avanzar en su madurez analítica.

Estos niveles ayudan a las empresas a identificar su estado actual de madurez analítica y trazar un camino para su desarrollo y mejora continua.

Tipos de Análisis:

- **Descriptivo:** ¿Qué ha ocurrido?
- **Predictivo:** ¿Qué podría ocurrir?
- **Prescriptivo:** ¿Qué deberíamos hacer?

Análisis Descriptivo Multivariable

Referencias:

- Applied multivariate statistics

El análisis descriptivo multivariable implica la exploración y el análisis de múltiples variables para entender las interrelaciones y cómo estas influyen en las decisiones empresariales. Este tipo de análisis es fundamental para comprender los contextos en los que operan los KPIs y cómo diferentes factores se influyen mutuamente.

Técnicas de Clustering

¿Qué es el Clustering? El clustering es una técnica de aprendizaje automático no supervisado que agrupa un conjunto de objetos de tal manera que los objetos en el mismo grupo (o cluster) son más similares entre sí que con los de otros grupos. Es ampliamente utilizado para la segmentación de datos en diferentes grupos.

¿Cuándo se utiliza? El clustering se emplea para explorar la estructura de los datos, para resumirlos en grupos manejables, o como una etapa de procesamiento previo para otros

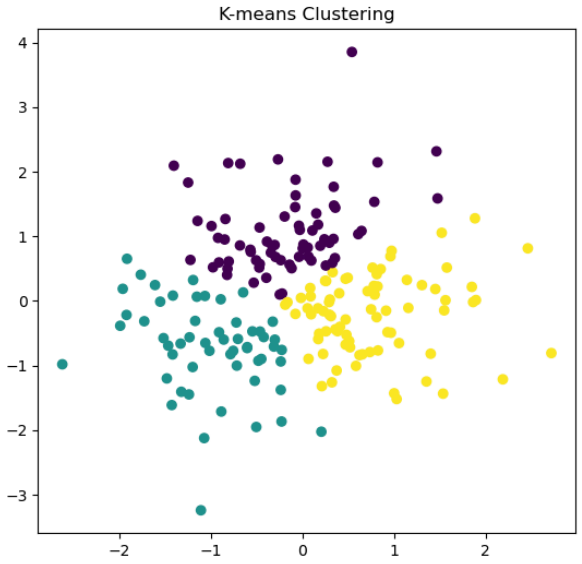
algoritmos. Es común en análisis de mercados, organización de computadoras en redes, clasificación de genes en biología, y en cualquier área que requiera una segmentación del conjunto de datos.

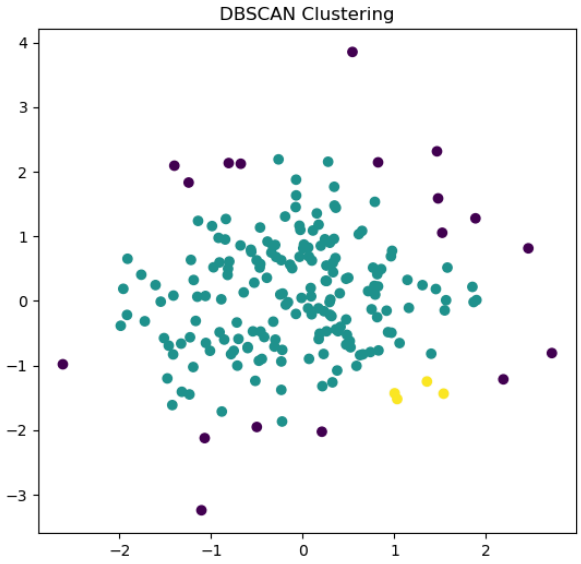
Beneficios del Clustering:

- **Descubrimiento de patrones:** Identifica agrupaciones y patrones desconocidos en conjuntos de datos.
- **Segmentación útil:** Facilita la segmentación de la base de clientes, productos o cualquier otra entidad analítica en grupos distintos para estrategias de targeting más específicas.
- **Optimización de recursos:** Permite la asignación eficiente de recursos al identificar y categorizar grupos con características similares.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans, DBSCAN
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(42)
data = pd.DataFrame({
    'Feature1': np.random.randn(200),
    'Feature2': np.random.randn(200)
})

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# K-means clustering
kmeans = KMeans(n_clusters=3)
kmeans_labels = kmeans.fit_predict(scaled_data)

# DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(scaled_data)

# Plotting the clusters
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# K-means plot
ax1.scatter(data['Feature1'], data['Feature2'], c=kmeans_labels, cmap='viridis')
ax1.set_title('K-means Clustering')

# DBSCAN plot
ax2.scatter(data['Feature1'], data['Feature2'], c=dbscan_labels, cmap='viridis')
ax2.set_title('DBSCAN Clustering')

plt.show()
```

```
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```



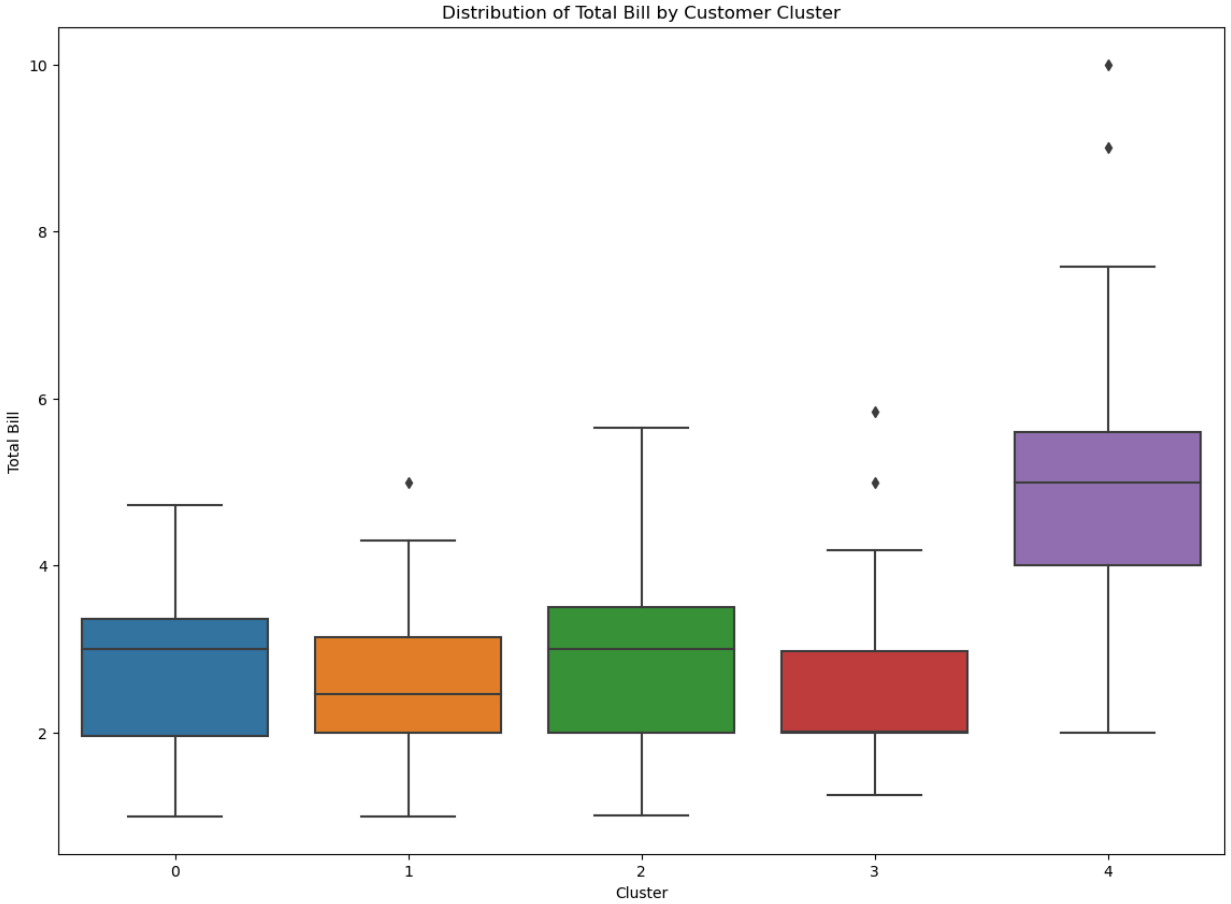


In [6]:

```python
# Importar librerías necesarias
import seaborn as sns
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn import set_config
set_config(working_memory=1024)

# Cargar el conjunto de datos 'tips' de Seaborn
data = sns.load_dataset('tips')

# Convertir variables categóricas en variables dummies
categorical_features = ['sex', 'smoker', 'day', 'time']
data_encoded = pd.get_dummies(data, columns=categorical_features, drop_first=True)

# Preparar los datos seleccionando características relevantes y normalizándolos
features = ['total_bill', 'tip', 'size'] + list(data_encoded.columns[5:])  # Incluir c
x = data_encoded.loc[:, features].values
x = StandardScaler().fit_transform(x)  # Normalización de las características

# Aplicar K-means clustering para identificar segmentos de clientes
kmeans = KMeans(n_clusters=5, random_state=42)
labels = kmeans.fit_predict(x)

# Agregar las etiquetas del cluster al DataFrame original para análisis
data_encoded['Cluster'] = labels

# Visualizar los resultados del clustering
plt.figure(figsize=(14, 10))

# Create a boxplot to visualize the distribution of 'total_bill' for each 'Cluster'
```

```
sns.boxplot(x='Cluster', y='tip', data=data_encoded)

# Set labels and title
plt.xlabel('Cluster')
plt.ylabel('Total Bill')
plt.title('Distribution of Total Bill by Customer Cluster')

# Display the plot
plt.show()
```

```
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [4]:

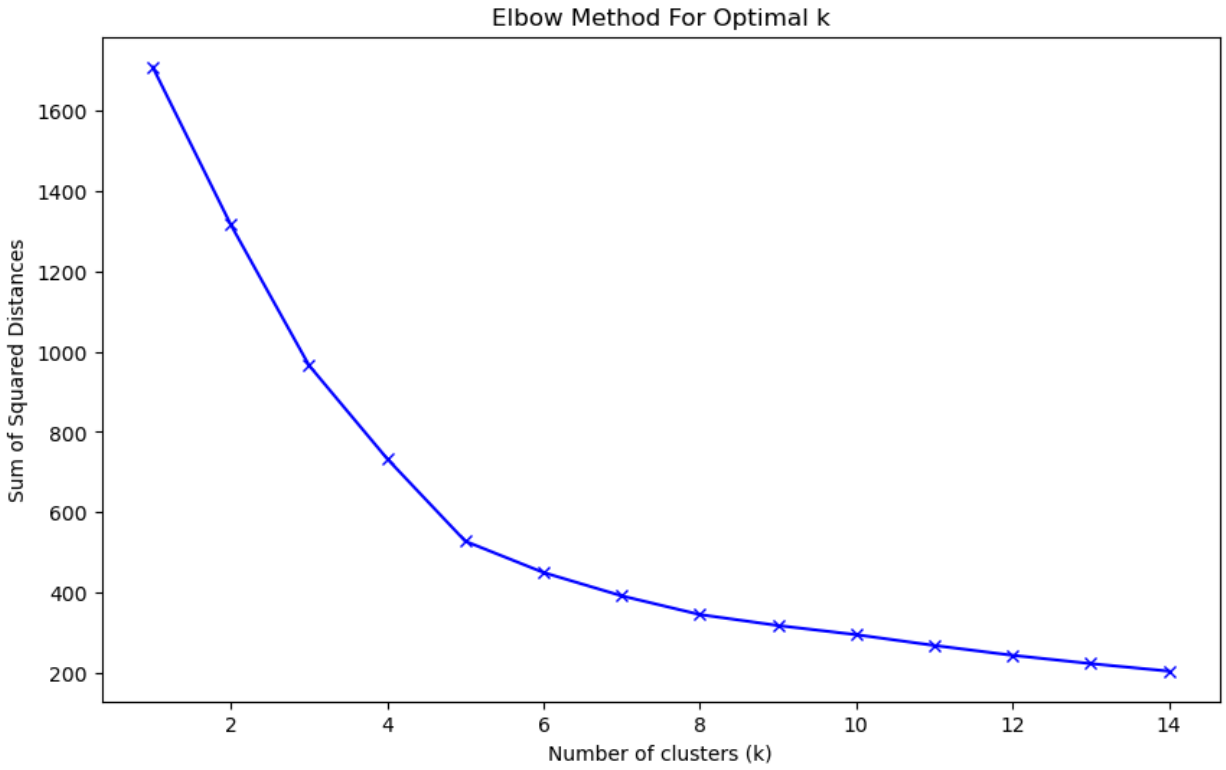
```
# Determinar el número óptimo de clusters usando el método del codo
sum_of_squared_distances = []
K = range(1, 15)  # Ajuste el rango según sea necesario
for k in K:
    km = KMeans(n_clusters=k, random_state=42)
    km = km.fit(x)
    sum_of_squared_distances.append(km.inertia_)

# Plot the Elbow curve
plt.figure(figsize=(10, 6))
plt.plot(K, sum_of_squared_distances, 'bx-')
```

```
plt.xlabel('Number of clusters (k)')
plt.ylabel('Sum of Squared Distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

```
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
```

```
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
```

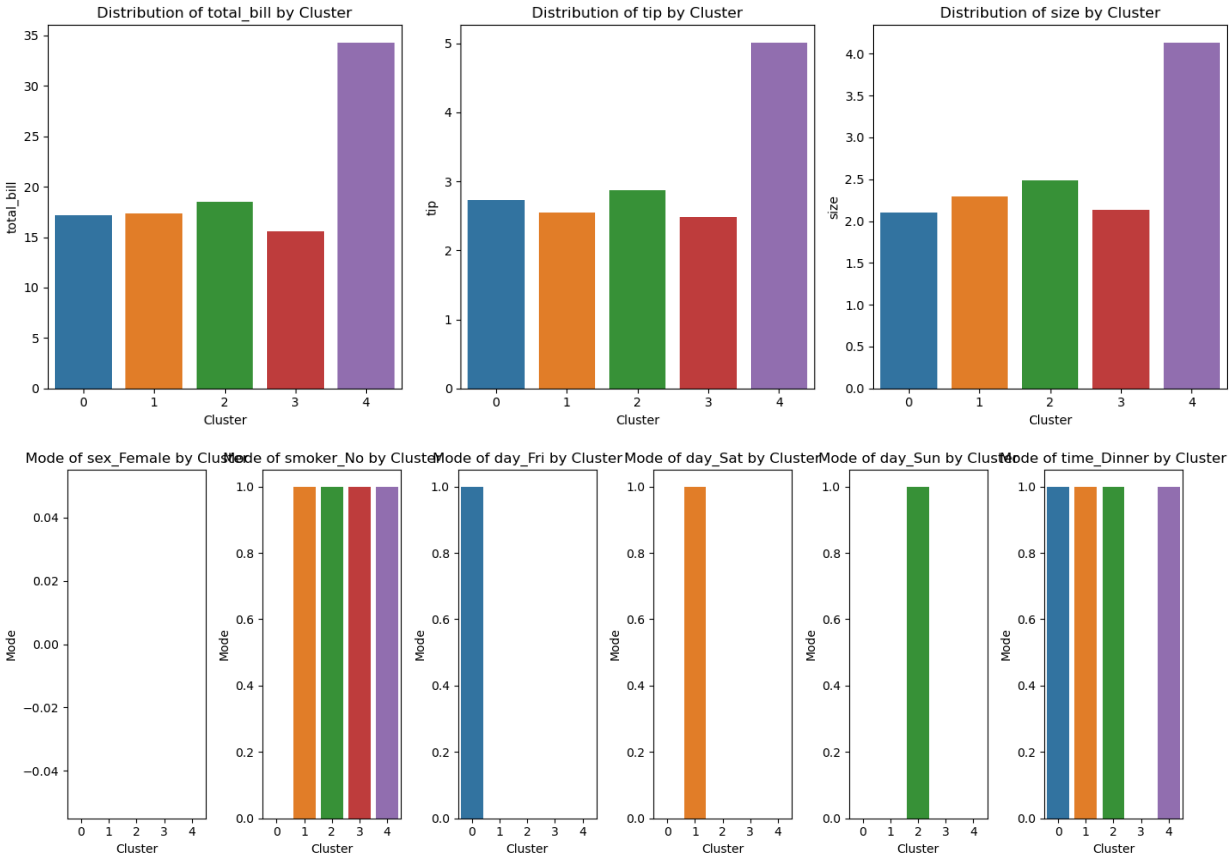
```
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\luism\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```



In [7]:

```
# Define a function to summarize the cluster characteristics
def summarize_clusters(data):
    # Numerical features
    numerical_features = ['total_bill', 'tip', 'size']  # Modify this list based on yo
    # Categorical features (assuming dummy variables were created)
    categorical_features = [col for col in data.columns if col.startswith('sex_') or c

    # Summary DataFrame for numerical features
    cluster_summary_numerical = data.groupby('Cluster')[numerical_features].mean()

    # Summary DataFrame for categorical features
    cluster_summary_categorical = data.groupby('Cluster')[categorical_features].agg(la

    return cluster_summary_numerical, cluster_summary_categorical

# Generate the summary for each cluster
cluster_summary_numerical, cluster_summary_categorical = summarize_clusters(data_encod

# Plotting
def plot_cluster_summary(cluster_summary_numerical, cluster_summary_categorical):
    # Plot numerical summaries
    fig, axes = plt.subplots(nrows=1, ncols=len(cluster_summary_numerical.columns), fi
    for i, col in enumerate(cluster_summary_numerical.columns):
        sns.barplot(x=cluster_summary_numerical.index, y=col, data=cluster_summary_num
        axes[i].set_title(f'Distribution of {col} by Cluster')
        axes[i].set_xlabel('Cluster')
        axes[i].set_ylabel(col)
```

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

    # Plot categorical summaries
    fig, axes = plt.subplots(nrows=1, ncols=len(cluster_summary_categorical.columns),
    for i, col in enumerate(cluster_summary_categorical.columns):
        sns.barplot(x=cluster_summary_categorical.index, y=col, data=cluster_summary_c
        axes[i].set_title(f'Mode of {col} by Cluster')
        axes[i].set_xlabel('Cluster')
        axes[i].set_ylabel('Mode')

    plt.tight_layout()
    plt.show()

# Call the plotting function
plot_cluster_summary(cluster_summary_numerical, cluster_summary_categorical)
```



Análisis de Componentes Principales (PCA)

¿Qué es PCA? El Análisis de Componentes Principales (PCA por sus siglas en inglés) es una técnica estadística utilizada para reducir la dimensionalidad de un conjunto de datos mientras se conserva la mayor cantidad posible de su variabilidad. PCA transforma las variables originales en un nuevo conjunto de variables, llamadas componentes principales, que son ortogonales entre sí y capturan la máxima varianza posible en los datos.

¿Cuándo se utiliza? PCA se utiliza frecuentemente en contextos donde se tienen grandes conjuntos de datos con muchas variables intercorrelacionadas. Es útil para simplificar los datos

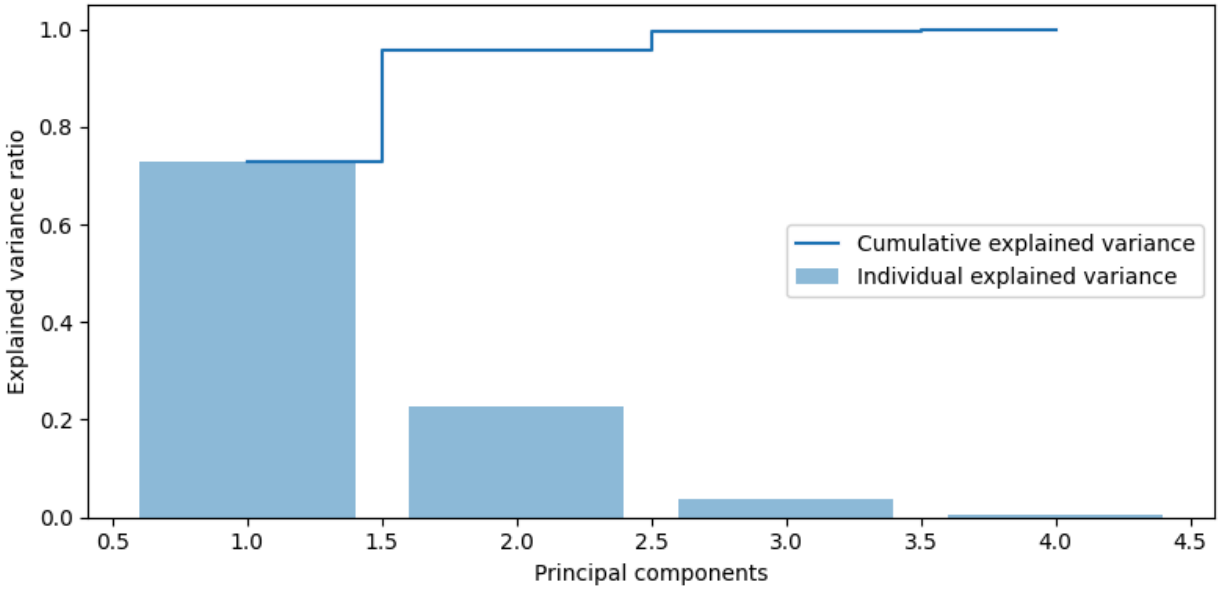
antes de realizar análisis más complejos, como en la modelización predictiva, o para visualizar patrones subyacentes que no son fácilmente identificables en datos multidimensionales.

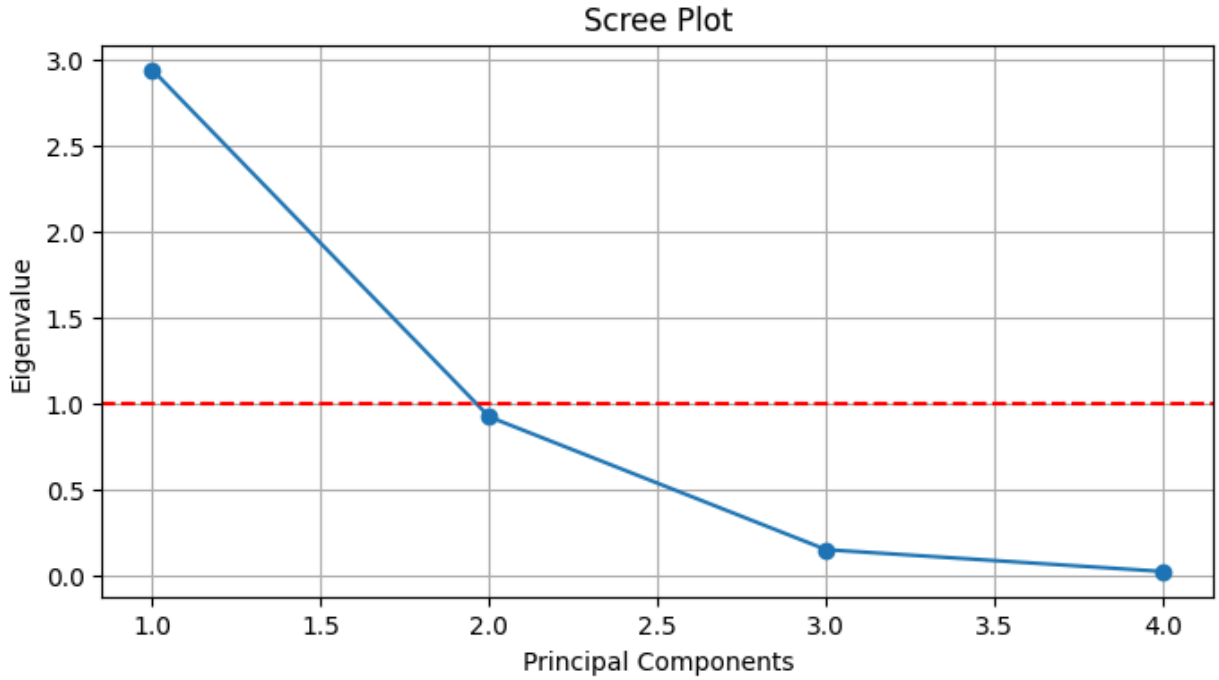
Beneficios de PCA:

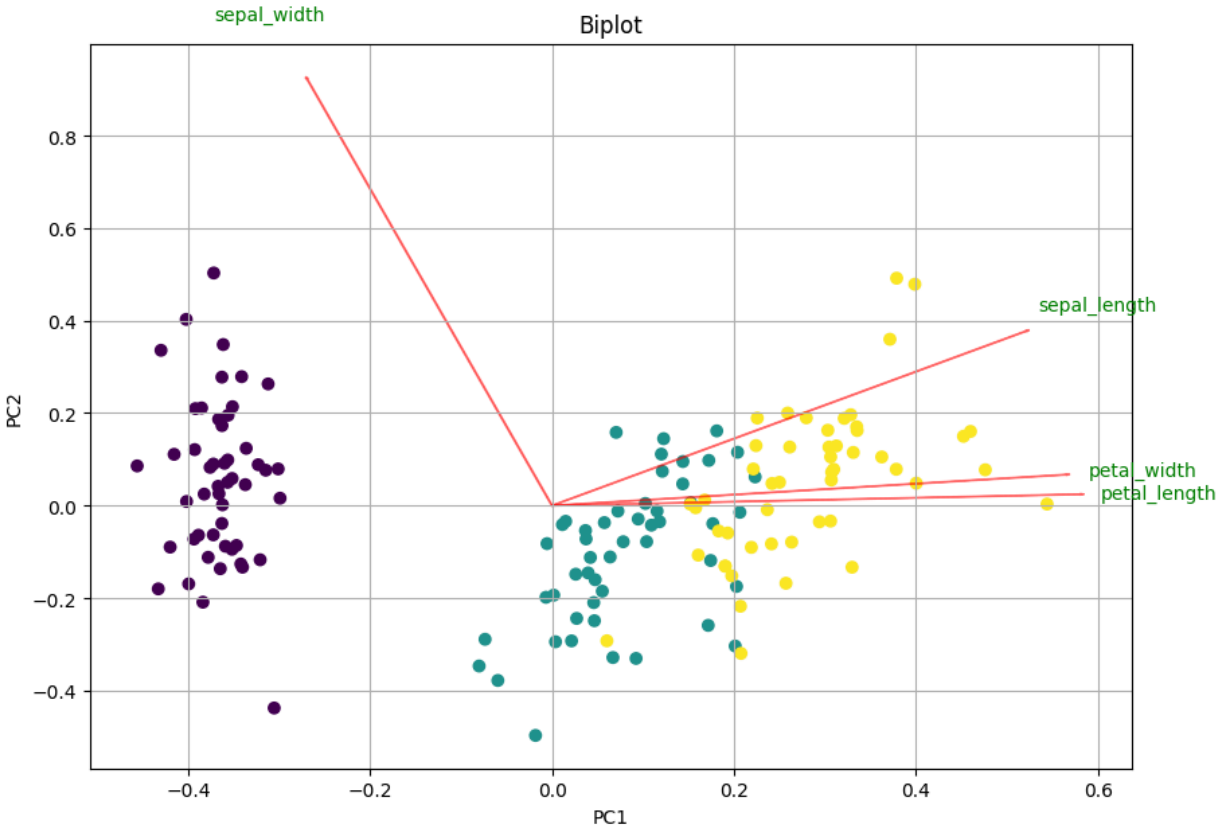
- **Reducción de la complejidad:** Simplifica la visualización y análisis de los datos.
- **Eliminación de la multicolinealidad:** Ayuda en la interpretación de los factores que influyen en el conjunto de datos al eliminar la redundancia.
- **Mejora de la eficiencia algorítmica:** Reduce el tiempo y los recursos computacionales necesarios para procesar los datos.

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import pandas as pd

# Load dataset
data = sns.load_dataset('iris')
features = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
x = data.loc[:, features].values

# Standardizing the features
x = StandardScaler().fit_transform(x)

# PCA
pca = PCA(n_components=4)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents, columns = ['PC1', 'PC2', 'PC3',

# Variance Explained Plot
plt.figure(figsize=(8, 4))
plt.bar(range(1, pca.n_components_ + 1), pca.explained_variance_ratio_, alpha=0.5, ali
plt.step(range(1, pca.n_components_ + 1), np.cumsum(pca.explained_variance_ratio_), wh
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal components')
plt.legend(loc='best')
plt.tight_layout()
plt.show()

# Scree Plot
plt.figure(figsize=(8, 4))
plt.plot(range(1, pca.n_components_ + 1), pca.explained_variance_, 'o-')
plt.title('Scree Plot')
plt.xlabel('Principal Components')
plt.ylabel('Eigenvalue')
plt.axhline(y=1, color='r', linestyle='--')
plt.grid(True)
plt.show()

# Biplot
def biplot(score, coeff, labels=None):
    xs = score[:,0]
    ys = score[:,1]
    scalex = 1.0/(xs.max() - xs.min())
    scaley = 1.0/(ys.max() - ys.min())
```

```
    plt.scatter(xs * scalex, ys * scaley, c=data['species'].astype('category').cat.coc
    for i in range(coeff.shape[0]):
        plt.arrow(0, 0, coeff[i,0], coeff[i,1], color='r', alpha=0.5)
        if labels is None:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, "Var"+str(i+1), color='g', h
        else:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, labels[i], color='g', ha='ce
    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.grid()

# Call the biplot
plt.figure(figsize=(10, 7))
biplot(principalComponents[:, :2], np.transpose(pca.components_[0:2, :]), labels=featu
plt.title('Biplot')
plt.show()
```







Beneficios del PCA y Clustering

- **PCA (Análisis de Componentes Principales)**: Esta técnica ayuda a reducir la dimensionalidad de los datos, manteniendo la mayor parte de la información significativa. PCA facilitará la visualización de la relación entre diversas variables y destacará las más influyentes en la cuenta total pagada.

- **Clustering**: Al aplicar técnicas de clustering, como K-means, el restaurante puede identificar segmentos de clientes con comportamientos similares. Esta segmentación permite la personalización de ofertas y estrategias específicas dirigidas a grupos de clientes que demuestran patrones de gasto similares, optimizando así las campañas de marketing y mejorando la satisfacción del cliente.

Conclusiones

Equipo:

- Luis Fernando Márquez Bañuelos
- Ivanna Herrera Ibarra
- Ana Sofía Hinojosa Bale

Código

La primera función del código nos da nuestras variables numéricas y categóricas, después agrupa por cluster en ambos tipos de variables, y saca la moda para las variables categóricas y

la media para las variables numéricas. La otra función es para graficar los clusters tanto por variables numéricas como por categóricas, utilizando un ciclo for para hacer las distintas gráficas.

Clusters

En el caso de las variables numéricas, los clusters del 0 al 3 son muy similares entre sí, ya que su tamaño de mesa, cuenta total y propina promedio son muy similares entre sí. Por otro lado, el cluster 4 tiene un tamaño de mesa, una cuenta total y una propina promedio más alta que el resto de los clusters.

Distribución de variables categóricas

En la variable de sexo nos muestra que en todos los clusters hay más hombres que mujeres. También nos dice que el cluster 0 tiene mayor cantidad de fumadores, mientras que en los otros la mayoría no fuma. El cluster 0 en su mayoría van los viernes, el cluster 1 en sábado y el 2 en domingo. Por último, el cluster 3 su mayoría va en el lunch mientras que los demás la mayoría va en la cena.

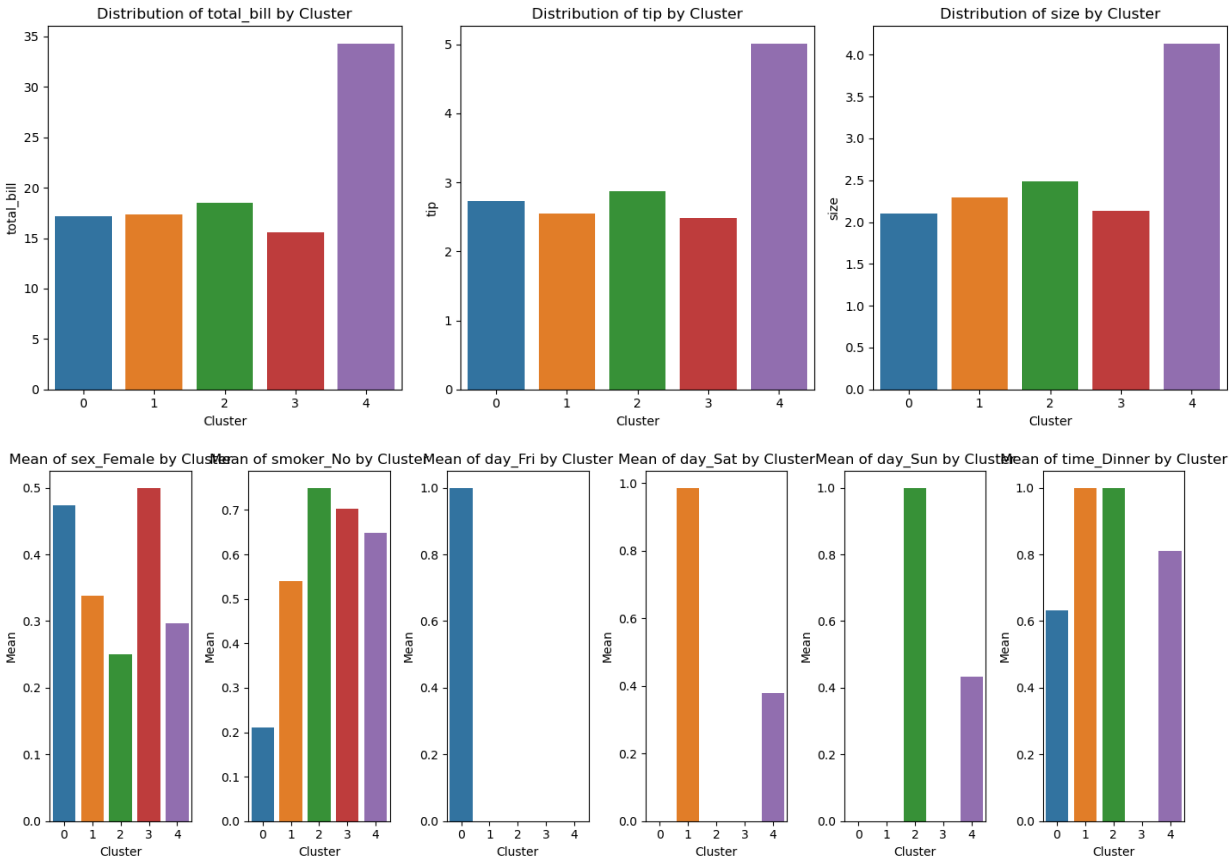
Nombres de los clusters

- Cluster 0: Pareja de hombres (mayoría) fumadores que van los viernes a cenar.
- Cluster 1: Pareja hombres (mayoría) no fumadores que van a cenar los sábados.
- Cluster 2: De 2 a 3 hombres (mayoría) no fumadores que cenan los domingos.
- Cluster 3: Pareja de hombres (mayoría) no fumadores que van al lunch.
- Cluster 4: Hombres (mayoría) no fumadores que van a cenar en mesas de 4 personas.

Punto Extra

In [12]:
```
def summarize_clusters(data):
    # Numerical features
    numerical_features = ['total_bill', 'tip', 'size']  # Modify this list based on yo
    # Categorical features (assuming dummy variables were created)
    categorical_features = [col for col in data.columns if col.startswith('sex_') or c

    # Summary DataFrame for numerical features
    cluster_summary_numerical = data.groupby('Cluster')[numerical_features].mean()

    # Summary DataFrame for categorical features
    cluster_summary_categorical = data.groupby('Cluster')[categorical_features].mean()

    return cluster_summary_numerical, cluster_summary_categorical

# Generate the summary for each cluster
cluster_summary_numerical, cluster_summary_categorical = summarize_clusters(data_encod

# Plotting
def plot_cluster_summary(cluster_summary_numerical, cluster_summary_categorical):
    # Plot numerical summaries
    fig, axes = plt.subplots(nrows=1, ncols=len(cluster_summary_numerical.columns), fi
    for i, col in enumerate(cluster_summary_numerical.columns):
```

```
        sns.barplot(x=cluster_summary_numerical.index, y=col, data=cluster_summary_num
        axes[i].set_title(f'Distribution of {col} by Cluster')
        axes[i].set_xlabel('Cluster')
        axes[i].set_ylabel(col)

    plt.tight_layout()
    plt.show()

    # Plot categorical summaries
    fig, axes = plt.subplots(nrows=1, ncols=len(cluster_summary_categorical.columns),
    for i, col in enumerate(cluster_summary_categorical.columns):
        sns.barplot(x=cluster_summary_categorical.index, y=col, data=cluster_summary_c
        axes[i].set_title(f'Mean of {col} by Cluster')
        axes[i].set_xlabel('Cluster')
        axes[i].set_ylabel('Mean')

    plt.tight_layout()
    plt.show()

# Call the plotting function
plot_cluster_summary(cluster_summary_numerical, cluster_summary_categorical)
```



Equipo:

- Luis Fernando Márquez Bañuelos
- Ivanna Herrera Ibarra
- Ana Sofía Hinojosa Bale

Para este caso en las variables categóricas, en lugar de utilizar la moda utilizamos el promedio, esto significa que lo graficado son los porcentajes de las diferentes variables categóricas para

cada cluster.

Se puede observar que los clusters 0 y 3 tiene casi la mitad de las mujeres lo cual es bastante y es información que no se apreciaba en las gráficas anteriores, esto es importante para saber en que clusters están las mujeres. El cluster 1 tiene poco más del 50% no fumadores por lo que casi la mitad fuman en ese cluster, del2 al 4 la gran mayoría no fuma. Otro hallazgo es que del cluster 4, que es de mesas grandes y gastan bastante, alrededor del 40% van los sábados y alrededor del 40% va en domingo. Por último, si coincide que una gran mayoría de todos los clusters los encuentras en la cena.

Al utilizar el promedio en estas variables nos dio mayor profundidad en las características de los clusters permitiendo así un análisis más completo. Con esto sabemos que el cluster 4, que es el más rentable, va la mayoría en sábados y domingos, y que la mayoría va a cenar, esto nos ayuda a ubicar cuando va nuestro cluster más rentable y prestar mayor atención en esos momentos. También vemos que los clusters 0 y 3 cuentan con bastantes mujeres, sabemos que el cluster 0 van todos en viernes, por lo que podemos esperar recibir bastantes mujeres los viernes y contruir estrategias a partir de esto. Del cluster 3 al estar en 0 en los días y cena, sabemos que acuden los jueves a la hora del lunch, por lo que ese día y hora también esperamos tener una cantidad de mujeres significativa.