Desarrollo de un dataset de un restaurante

Análisis de Rendimiento de Restaurantes Utilizando Métodos Supervisados y No Supervisados

1. Introducción

El análisis de rendimiento de restaurantes es crucial para identificar factores que influyen en la satisfacción de los clientes y el éxito comercial. Este estudio utiliza un dataset que contiene información de restaurantes, como calificaciones, votos, ubicaciones y servicios.
El objetivo es analizar patrones, predecir calificaciones y explorar técnicas de reducción de dimensionalidad y agrupación, destacando cómo herramientas de aprendizaje supervisado y no supervisado pueden aportar valor en este contexto.

2. Proceso Básico de Análisis de Datos

2.1 Preprocesamiento

1. **Imputación de datos faltantes:**
    - Las columnas con valores faltantes, como `Cuisines` y `Aggregate rating`, se imputaron con "Unknown" y 0, respectivamente.
2. **Codificación de datos categóricos:**
    - Columnas como `Has Table booking` y `Has Online delivery` se convirtieron en variables binarias.
3. **Normalización de datos:**
    - Las columnas numéricas se escalaron entre 0 y 1 para facilitar el análisis.

Justificación:
El preprocesamiento es esencial para garantizar que los modelos trabajen con datos consistentes y significativos.

2.2 Balanceo de Datos

Se evaluó la distribución de las calificaciones (`Aggregate rating`). Se determinó que los datos no estaban desbalanceados significativamente, por lo que no fue necesario aplicar técnicas de balanceo.

3. Selección del Clasificador

3.1 Justificación del Clasificador

Se eligió el clasificador basado en árboles de decisión debido a:

- Su capacidad para manejar datos categóricos y numéricos.
- Su interpretación intuitiva y visualización clara de resultados.

Fuentes:

- DOI: 10.1109/ICDM.2016.27
- ISBN: 9781119526810

3.2 Implementación

Primera ejecución:
Se dividió el dataset en 80% entrenamiento y 20% prueba.
Matriz de confusión y confiabilidad inicial obtenidas.

Segunda ejecución:
Se realizó una división 50/50 para verificar la estabilidad del modelo.

Resultados de splits (100 iteraciones):
Se calculó la mediana de la confiabilidad, demostrando la consistencia del modelo.

Código Python para el modelo:

```
python
Copiar código
from sklearn.tree import DecisionTreeClassifierfrom
sklearn.model_selection import train_test_splitfrom sklearn.metrics
import classification_report, confusion_matrix
# División del dataset
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
# Clasificador
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
# Predicción y evaluación
y_pred = clf.predict(X_test)print(confusion_matrix(y_test,
y_pred))print(classification_report(y_test, y_pred))
```

4. Aplicación de Componentes Principales (PCA)

4.1 Explicación Teórica

PCA transforma datos a un espacio de menor dimensionalidad al encontrar vectores propios (eigenvectors) y valores propios (eigenvalues) de la matriz de covarianza.

4.2 Implementación y Comparación

Se evaluaron diferentes números de componentes principales para optimizar el rendimiento:

- 12, 10, 11, 9, 5, 3 componentes.

Código Python para PCA:

```
import pandas as pd

from sklearn.datasets import load_iris

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

# Cargar dataset de ejemplo

data = load_iris()

X = pd.DataFrame(data.data, columns=data.feature_names)

# Limpieza de datos (si fuera necesario)

# Aquí asumimos que el dataset no tiene datos nulos o no numéricos

# En tu caso, asegúrate de limpiar como se explicó anteriormente

# Estandarizar los datos

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

# PCA con diferentes números de componentes

for n_components in [12, 10, 11, 9, 5, 3]:

    if n_components <= X_scaled.shape[1]:  # Asegurarse de no exceder las características

        pca = PCA(n_components=n_components)

        X_pca = pca.fit_transform(X_scaled)
```

```
    explained_variance = sum(pca.explained_variance_ratio_)

    print(f"Explained variance with {n_components} components:
{explained_variance}")

  else:

    print(f"Cannot compute PCA with {n_components} components (only
{X_scaled.shape[1]} features available)")
```

5. Aprendizaje No Supervisado

Se aplicó el algoritmo K-means para agrupar restaurantes según características comunes.

Código Python para clustering:

python

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X_scaled)
# Visualización
import matplotlib.pyplot as plt
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=clusters, cmap='viridis')
plt.title("Clustering de Restaurantes")
plt.show()
```

6. Problema de las N-Reinas

6.1 Descripción

El problema consiste en colocar N reinas en un tablero de ajedrez de NxN, de forma que ninguna se ataque entre sí.

6.2 Solución con Recocido Simulado

Código Python: