Tecnológico de Monterrey

*Maestría en Inteligencia Artificial Aplicada (MNA-V)*

**Trimestre Abr - Jul 2024**

# Análisis de Grandes Volúmenes de Datos

## Avance 2 - Sistema de Recomendación

**Equipo #11**


| | |
|---|---|
| **Daniel Guzmán Ávila** | **A00781387** |
| **Julio Cesar Ruiz Marks** | |
| **Alicia Sanchez Carmona** | **A01652134** |


**Profesor Titular: Dr. Nestor Velasco**

**26.05.24**

## 1) Métricas de evaluación

### 1.1 RMSE

El Root Mean Square Error (RMSE) es una métrica que se utiliza para medir la diferencia entre los valores reales y los de un modelo. Se utiliza esta métrica como medida directa de la precisión de un modelo. Penaliza los errores grandes y la métrica está en la misma escala que los valores originales.

### 1.2 Precisión

Esta métrica evalúa la calidad de las recomendaciones hechas, en este caso en las primeras 10 posiciones. Evalúa cuántos de los primeros 10 elementos son relevantes a la predicción. Esta métrica mide la relevancia en las primeras posiciones para enfocarnos en las recomendaciones inmediatas del usuario, que podría no explorar mucho la lista. Es de gran utilidad para comparar este modelo con otros modelos de recomendación.

### 1.3 MAE

MAE es una métrica de evaluación que mide la media de los errores absolutos entre las predicciones y calificaciones reales. Es una métrica fácil de interpretar, a diferencia del RMSE es menos sensible a los outliers así que es un complemento de otra métrica que elegimos que sería RMSE

## 2) Experimentación con algoritmo de recomendación básico

El algoritmo de recomendación básico utilizado es la similitud del coseno, la cual es una métrica utilizada para medir cuán similares son dos vectores en un espacio de características, independientemente de su magnitud.

```
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim = cosine_similarity(games_features, games_features)

print(f"Las dimensiones de similaridad coseno de las caracteristicas
de nuestra matriz de similitud son: {cosine_sim.shape}")
```

Las dimensiones de similaridad coseno de las características de nuestra matriz de similitud son: (149, 149)

```
numero_recomendaciones=10

sim_scores = list(enumerate(cosine_sim[idx]))

sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
```

```
sim_scores = sim_scores[1:(numero_recomendaciones+1)]

similar_games = [i[0] for i in sim_scores]

print(f"Por que jugaste {title}, te puede interesar
{numero_recomendaciones}")

promedios_juego['app_name'].iloc[similar_games]
```

Por que jugaste Counter-Strike: Source, te puede interesar 10

| | |
|---|---|
| 1 | Counter-Strike: Source |
| 2 | Half-Life 2: Episode Two |
| 7 | BioShock Infinite |
| 12 | Saints Row: The Third |
| 33 | Getting Over It with Bennett Foddy |
| 40 | The Binding of Isaac: Rebirth |
| 70 | Down To One |
| 74 | Tom Clancy's Rainbow Six Siege |
| 77 | DOOM |
| 78 | Dead by Daylight |

### 3) Descripción del algoritmo de recomendación avanzado

Un sistema de recomendación tiene como objetivo recomendar uno o más elementos a los usuarios del sistema. En nuestro caso en particular estamos viendo recomendaciones de videojuegos. Hay dos maneras principales para abordar los sistemas de recomendaciones, una es basada en el contenido, utiliza propiedades de ambos los usuarios y los elementos recomendados. Otra manera para abordar las recomendaciones es filtrado colaborativo, utiliza información implícita de una matriz de evaluaciones por parte de los usuarios a los elementos, en el caso de SVD es un sistema de recomendación colaborativo.

SVD es Singular Value Decomposition es una técnica de factorización matricial utilizada para la reducción de dimensionalidad. Este algoritmo busca aproximar la matriz de utilidad que contiene las calificaciones de los usuarios utilizando una matriz de menor rango.

La manera en la que implementamos el algoritmo es de esta manera

1. Creamos la matriz de utilidad

```
# Crear la matriz de utilidad
UtMx_games = pd.pivot_table(game, values='recommended', index='review_id', columns='app_name')
UtMx_games.fillna(0, inplace=True)

print('Dimensión de la matriz de Utilidad:')
print('(usuarios, juegos) =', UtMx_games.shape)
print(UtMx_games.head())

utility_matrix = UtMx_games.values
```

2. Aplicamos la transformación SVD para obtener la matriz de las variables latentes de los juegos en relación con las recomendaciones.

```
nc_games = min(UtMx_games.shape) - 1

SVD_games = TruncatedSVD(n_components=nc_games)
SVD_games.fit(UtMx_games)

for j in range(nc_games):
    if SVD_games.explained_variance_ratio_[0:j].sum() > 0.90:
        break

N_games = j - 1

print('Total de valores singulares basados en la evaluación de los juegos:', nc_games)
print('Valor de truncamiento al 90% de dicha variabilidad:', N_games)
```

3. Se utiliza el método svds con la matriz de utilidad para crear las predicciones

```
u, s, vt = svds(utility_matrix, k=N_games)

s_diag_matrix = np.diag(s)
predicted_ratings = np.dot(np.dot(u, s_diag_matrix), vt)

predicted_ratings_df = pd.DataFrame(predicted_ratings, columns=UtMx_games.columns)

print(predicted_ratings_df.head())
```

4. Utilizando la matriz de correlación Pearson conseguimos los 10 juegos a recomendar teniendo como referencia 1 juego.

```
# Matriz de correlación de Pearson
corr_mat = np.corrcoef(predicted_ratings.T)

# Juego de referencia
juego_de_referencia = "Portal 2"
idx_juego = list(UtMx_games.columns).index(juego_de_referencia)
corr_juego = corr_mat[idx_juego]

# Correlaciones positivas
idx = (corr_juego > 0)
mejores_sim_games = list()
for i in range(len(UtMx_games.columns[idx])):
    mejores_sim_games.append((corr_juego[idx][i], UtMx_games.columns[idx][i]))

print('Total de similaridades positivas encontradas:', len(mejores_sim_games))

print('Algunos de los resultados encontrados:')
print(mejores_sim_games[0:10])

mejores_sim_games_ordenadas = sorted(mejores_sim_games, key=lambda x: x[0], reverse=True)


# Desplegamos las 10 mejores similitudes encontradas de manera descendente:
print('Similitudes con base a la evaluación del servicio con mayores valores de correlación:')
for k in range(1,11):
  print('%d> %s' % (k, mejores_sim_games_ordenadas[k]))
```

**Referencias.**


- Maklin, C. (2022, August 9). Model Based Collaborative Filtering — SVD - Cory Maklin - Medium. *Medium*. https://medium.com/@corymaklin/model-based-collaborative-filtering-svd-19859c764cee
    - 
- *Oracle® Fusion Cloud EPM Trabajo con Planning*. (n.d.). https://docs.oracle.com/cloud/help/es/pbcs_common/PFUSU/insights_metrics_RMSE.htm#PFUSU-GUID-FD9381A1-81E1-4F6D-8EC4-82A6CE2A6E74
- Likebupt. (2022, July 17). *Train SVD Recommender: Component Reference - Azure Machine Learning*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/train-svd-recommender?view=azureml-api-2