```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Sample movie dataset
movies_data = {
    'movie_id': [1, 2, 3, 4, 5],
    'title': ['Inception', 'The Matrix', 'Interstellar', 'The Prestige', 'Memento'],
    'genres': ['Action Sci-Fi', 'Action Sci-Fi', 'Adventure Drama Sci-Fi',
'Drama Mystery Sci-Fi', 'Mystery Thriller'],
    'description': [
        'A thief steals corporate secrets through dream-sharing technology.',
        'A hacker discovers the world is a simulation.',
        'A team travels through a wormhole in space.',
        'Magicians compete with dangerous tricks.',
        'A man with short-term memory loss uses notes to hunt a killer.'
    ]
}

# Create DataFrame
movies_df = pd.DataFrame(movies_data)
```

```
# Combine genres and descriptions for content
movies_df['content'] = movies_df['genres'] + " " + movies_df['description']

# Vectorize the content
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(movies_df['content'])

# Compute similarity matrix
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Helper function: get recommendations
def get_recommendations(title, top_n=3):
    idx = movies_df.index[movies_df['title'] == title].tolist()[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:top_n+1]
    movie_indices = [i[0] for i in sim_scores]
    return movies_df[['title', 'genres']].iloc[movie_indices]

# Simulate a user profile (likes Sci-Fi and mystery)
user_preferences = "Sci-Fi mystery hacker dream memory"
```

```
# Vectorize the user profile
user_vec = tfidf.transform([user_preferences])

# Compute similarity between user and all movies
user_sim = cosine_similarity(user_vec, tfidf_matrix).flatten()

# Match top N movies for user
def recommend_for_user(user_sim, top_n=3):
    indices = np.argsort(user_sim)[::-1][:top_n]
    return movies_df[['title', 'genres']].iloc[indices]

# Display personalized recommendations
print("Personalized Recommendations:")
print(recommend_for_user(user_sim, top_n=3))

# Matchmaking system: recommend similar users (if user ratings were present, collaborative filtering could apply)
# Here we simulate one user profile and match to movies using AI (content similarity)

# Optional: Save model and data
# import joblib
```

```
# joblib.dump(tfidf, 'tfidf_vectorizer.pkl')

# movies_df.to_csv('movies.csv', index=False)
```