# Movie Recommendation System — Full Documentation + Complete Code

This document includes complete project explanation and ALL code extracted exactly as it exists in the notebook (no edits or additions).

1. Overview

---

This project builds a Movie Recommendation System using similarity-based techniques.

The notebook includes dataset loading, text processing, feature vectorization,

similarity computation, and recommendation generation.

2. Libraries Used

---

• pandas — dataset loading and processing

• numpy — numerical operations

• sklearn.feature_extraction.text — CountVectorizer or TfidfVectorizer

• sklearn.metrics.pairwise — cosine_similarity

• difflib — for matching user-input movie titles

• nltk — for stemming/tokenization (if used)

3. Workflow Steps

---

1. Load the movies dataset

2. Combine relevant movie features for text representation

3. Convert combined text into numerical vectors

4. Compute similarity matrix using cosine similarity

5. Accept movie name from user

6. Find closest matching title

7. Retrieve similarity scores

8. Recommend top similar movies

4. Important Variables

---

• movies / df — main dataset

• combined_features — processed textual representation

• vectorizer — TF-IDF or CountVectorizer

• feature_vectors — numeric movie embeddings

• similarity — cosine similarity matrix

• movie_name — user input

• close_match — difflib output

• recommendations — top similar movies

5. Notes

---

• Recommendation method is content■based

• difflib used for fuzzy title matching

• No missing imports detected

• Code included exactly as-is below

6. Complete Notebook Code

---

All code cells are extracted exactly as they appear in the notebook.

### *Code Cell 1*

```
import numpy as np
import pandas as pd

import difflib as dfl

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

### *Code Cell 2*

```
movies_data = pd.read_csv("/content/drive/MyDrive/Data Science/Projects/18 Movie Recommendation Syste
movies_data.head(2)
```

### *Code Cell 3*

```
#no. of rows and cols
movies_data.shape
```

### *Code Cell 4*

```
movies_data.info(0)
```

### *Code Cell 5*

```
movies_data.isnull().sum()
```

### *Code Cell 6*

```
selected_features = ['genres','keywords','tagline','cast','director']
print(selected_features)
```

### *Code Cell 7*

```
#replacing the null values with nill string
for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('')
```

### *Code Cell 8*

```
movies_data.isnull().sum()
```

### *Code Cell 9*

```
#combine the 5 selected features
combined_features = movies_data['genres'] + ' ' +  movies_data['keywords'] + ' ' +  movies_data['tag
```

### *Code Cell 10*

```
combined_features
```

### *Code Cell 11*

```
#converting text data to feature vectors
vectorizer = TfidfVectorizer()

feature_vectors = vectorizer.fit_transform(combined_features)
print(feature_vectors)
```

### *Code Cell 12*

```
#getting the similarity score using cosine similarity
similarity = cosine_similarity(feature_vectors)
```

```
print(similarity)
```

### *Code Cell 13*

```
#getting the movie name from user
movie_name = input("Enter movie name: ")
```

### *Code Cell 14*

```
#creating a list with all the movie names given in the dataset
list_of_all_titles = movies_data['title'].tolist()
print(list_of_all_titles)
```

### *Code Cell 15*

```
#finding the close match for the movie name given by the user
find_close_match = dfl.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)
```

### *Code Cell 16*

```
close_match = find_close_match[0]
print(close_match)
```

### *Code Cell 17*

```
#finding the index of the movie
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)
```

### *Code Cell 18*

```
#getting a list of similar movies
similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)
```

### *Code Cell 19*

```
len(similarity_score)
```

### *Code Cell 20*

```
#sorting the movies now based on score
sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)
print(sorted_similar_movies)
```

### *Code Cell 21*

```
#print the name of the movies based on the index
print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<10):
```

```
        print(i, '.', title_from_index)
        i+=1
```

### *Code Cell 22*

```
# (Empty code cell)
```