SmartPhoneMatch: A Content-Based Filtering System for Mobile Recommendations

Abstract

With the vast number of mobile phones on the market, consumers often face difficulty choosing the right device. This project presents a content-based recommendation system that suggests similar smartphones based on the specifications of a phone a user already likes. Using **TF-IDF vectorization** and **cosine similarity**, the model analyzes textual data like RAM, processor, display, and more to find and recommend closely related mobile phones. This approach ensures personalized suggestions based on content similarity rather than user behavior.

Objective

To build a recommendation system that:

- Understands the specifications of a given mobile phone.
- Finds similar phones based on feature similarity.
- Returns top-N mobile suggestions with details like name, price, and rating.

Dataset Description

The dataset used contains information about various mobile phones, including:

Column Name	Description
`name`	Mobile phone name and model
`price`	Market price of the phone (cleaned from ₹ and commas)
`ratings`	Average user rating
`ram`, `rom`, `display`, `processor`	Key specifications
`features`	Additional features like camera, battery, etc.

Data Cleaning & Preprocessing

1. **Price Cleaning:**
    a. Removed symbols like "₹", commas.
    b. Converted to float for numeric operations.
2. **Missing Values:**
    a. Dropped or filled missing values using `fillna()`.
3. **Textual Feature Engineering:**
    a. Combined multiple columns (`ram`, `rom`, `display`, `processor`, `features`) into one column `description`.
    b. This `description` column is the base for vectorization.

Modeling: Content-Based Filtering

What is Content-Based Filtering?

Content-based filtering recommends items based on **item similarity**, not user ratings. It analyzes the features of the items and matches them.

For example, if you search for "Samsung Galaxy M13", the model checks its features and finds other mobiles with similar specifications.

Feature Representation using TF-IDF

We used **TF-IDF Vectorization** to convert text data into numbers.

- **TF (Term Frequency):** How often a word appears.
- **IDF (Inverse Document Frequency):** Reduces weight of commonly used terms (like "phone", "mobile").

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df['description'])
```

- Now, `tfidf_matrix` is a numeric matrix where each row is a mobile, and each column is a word (feature), with a TF-IDF score.

Calculating Cosine Similarity

Cosine similarity measures how similar two vectors (mobiles) are:

- Formula:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| \times ||B||}$$

  - It ranges from `0` (completely different) to `1` (exactly same).

```
python

from sklearn.metrics.pairwise import cosine_similarity

cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

-
- The result is a square matrix: `cosine_sim[i][j]` gives similarity between mobile `i` and mobile `j`.

Recommendation Function

```
[ ]  def recommend_mobiles(mobile_name, top_n=10):
         try:
             idx = df[df['name'].str.contains(mobile_name, case=False, na=False)].index[0]
         except IndexError:
             return f"No mobile found matching: {mobile_name}"

         sim_scores = list(enumerate(cosine_sim[idx]))
         sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)[1:top_n+1]

         recommended_indices = [i[0] for i in sim_scores]
         return df.iloc[recommended_indices][['name', 'price', 'ratings']]
```

How the Function Works:

1. **Search** for the mobile in the `name` column using case-insensitive match.
2. **Get the index** of that mobile from the dataframe.
3. **Fetch similarity scores** using cosine similarity matrix.
4. **Sort the scores** in descending order and skip the first one (as it is the same mobile).
5. **Return top N similar mobiles**, showing their names, prices, and ratings.

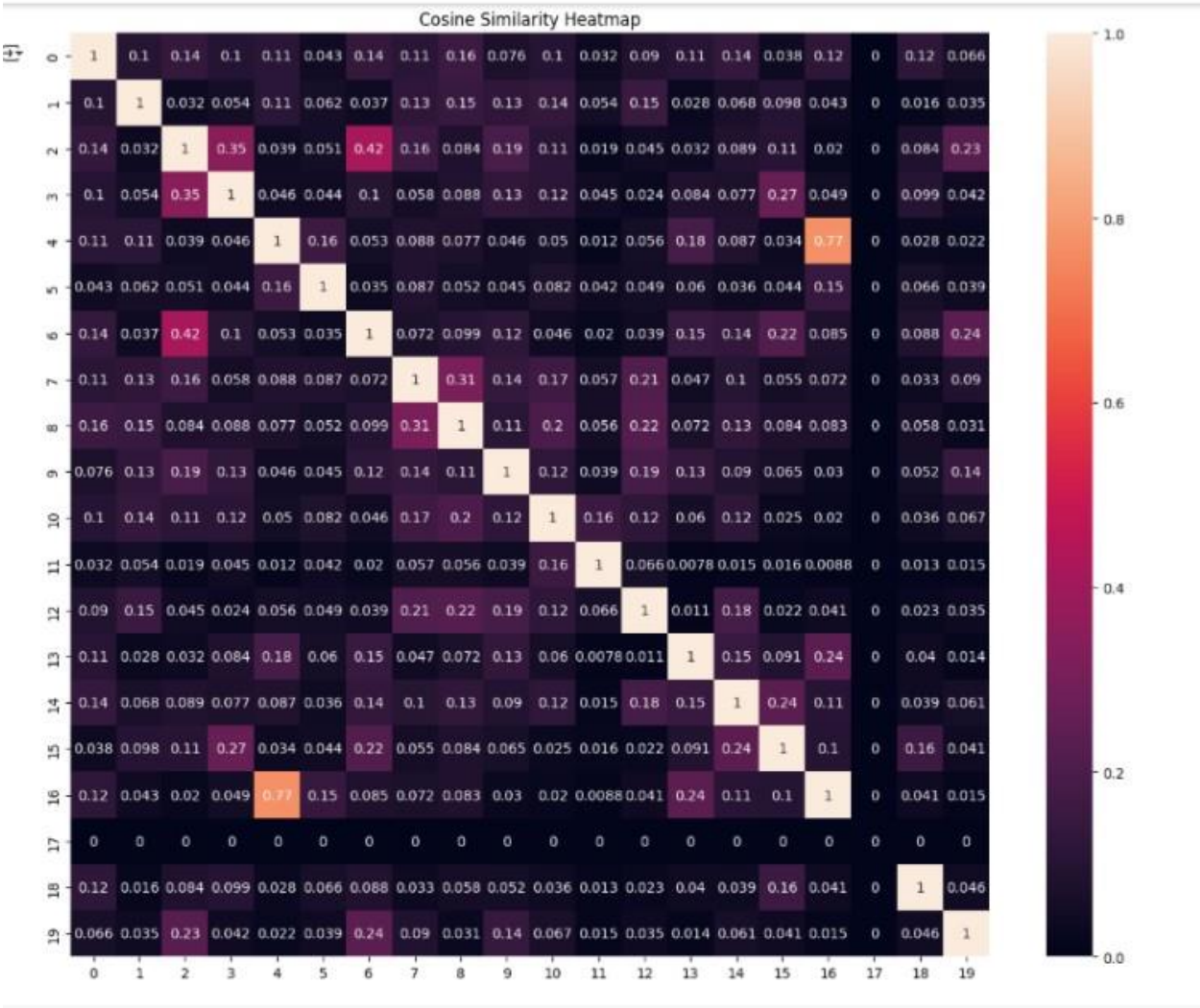
Example Output

```
user_input = input("Enter a mobile name: ")
print("\nTop Recommendations:\n")
print(recommend_mobiles(user_input))
```

```
Enter a mobile name: SAMSUNG

Top Recommendations:

                                              name  price  ratings
2480       SAMSUNG Galaxy A04 (Copper, 128 GB)  12799      4.0
1028        SAMSUNG Galaxy A04 (Black, 128 GB)  11990      4.0
1539        SAMSUNG Galaxy A04 (Copper, 64 GB)  11199      4.4
1654      SAMSUNG Galaxy A04e (Copper, 128 GB)  11499      4.1
2104  SAMSUNG Galaxy A04e (Light Blue, 128 GB)  11499      4.1
1451       SAMSUNG Galaxy A04e (Copper, 64 GB)   9999      4.2
911          SAMSUNG Galaxy A13 (Blue, 128 GB)  15499      4.1
193          SAMSUNG Galaxy A12 (Blue, 128 GB)  17999      4.2
1607        SAMSUNG Galaxy A13 (Black, 128 GB)  14999      4.1
2305        SAMSUNG Galaxy A13 (Peach, 128 GB)  16499      4.2
```


Visualization

```
import seaborn as sns
import matplotlib.pyplot as plt

sample_cosine = cosine_sim[:20, :20]
plt.figure(figsize=(12, 10))

sns.heatmap(sample_cosine, annot=True)
plt.title("Cosine Similarity Heatmap")
plt.tight_layout()
plt.show()
```



Conclusion

- We successfully built a content-based recommender that uses TF-IDF and cosine similarity.
- Users get relevant recommendations based on mobile specs.
- The model is scalable and can be improved with more feature data.

Future Enhancements

- Add **fuzzy matching** to handle typos in mobile names.
- Add **filters** like price range or specific brand.
- Deploy using **Flask/Streamlit** as a user-friendly web app.
- Convert to **Hybrid Filtering** by adding collaborative signals (user reviews, browsing behavior).

References

- Scikit-learn Documentation
- Cosine Similarity Wiki
- TF-IDF Guide