"Empowerment through quality technical education"

Dr D Y Patil School of Engineering

D Y Patil Knowledge City, Charholi (Bk), Via Lohegaon, Pune 412105

Affiliated to S. P. Pune University Pune, MSBTE, Approved by AICTE & DTE

(Accredited by NAAC, NABL & ISO 9001:2015 & 21001:2018 Certified Institute)

D Y PATIL GROUP

Department of Computer Engineering

Mini Project on DSBDA Laboratory

Members:

TE_A_81 Omkar Nandode (PRN No. 72034728K)
TE_A_76 Muskan Sawant (PRN No. 72034784L)

Title

Building a Movie Recommendation model in Python using Scikit-Learn.

Problem Definition:

To Develop a movie recommendation model using the scikit-learn library in python.

Objective:

The Objective of this recommendation system is to provide satisfactory movie recommendations to users while keeping the system user friendly i.e. by taking minimum input from users, It recommends the movie based on metadata of the movies and past user ratings.

Prerequisite:

Basic concepts of Python and Scikit-Learn library.

Software Requirements:

Python.
Library required: Pandas, Numpy, Difflib, AST, Scikit-Learn.

Outcomes:

Rightly recommended movie.

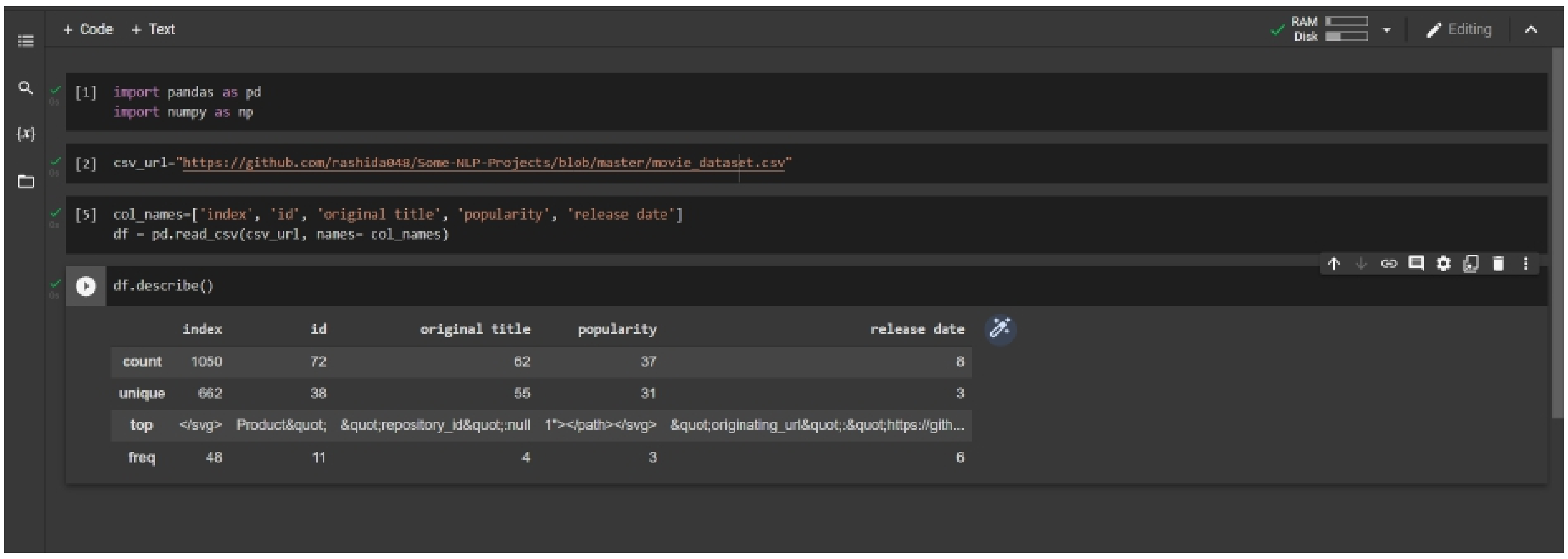
Data Set Description

The techniques that can be used to collect data are:

1. Explicit, where data are provided intentionally as an information (e.g. user's input such as movies rating)
2. Implicit, where data are provided intentionally but gathered from available data stream
(e.g. search history, clicks, order history, etc…)

Dataset

https://github.com/rashida048/Some-NLP-Projects/blob/master/movie_dataset.csv



```
[1] import pandas as pd
    import numpy as np
```

```
[2] csv_url="https://github.com/rashida048/Some-NLP-Projects/blob/master/movie_dataset.csv"
```

```
[5] col_names=['index', 'id', 'original title', 'popularity', 'release date']
    df = pd.read_csv(csv_url, names= col_names)
```

```
df.describe()
```

```
        index        id             original title    popularity                               release date
count    1050        72                         62            37                                          8
unique    662        38                         55            31                                          3
top    </svg>  Product&quot;  &quot;repository_id&quot;:null  1"></path></svg>  &quot;originating_url&quot;:&quot;https://gith...
freq       48        11                          4             3                                          6
```

Theory

-> Scikit-Learn:

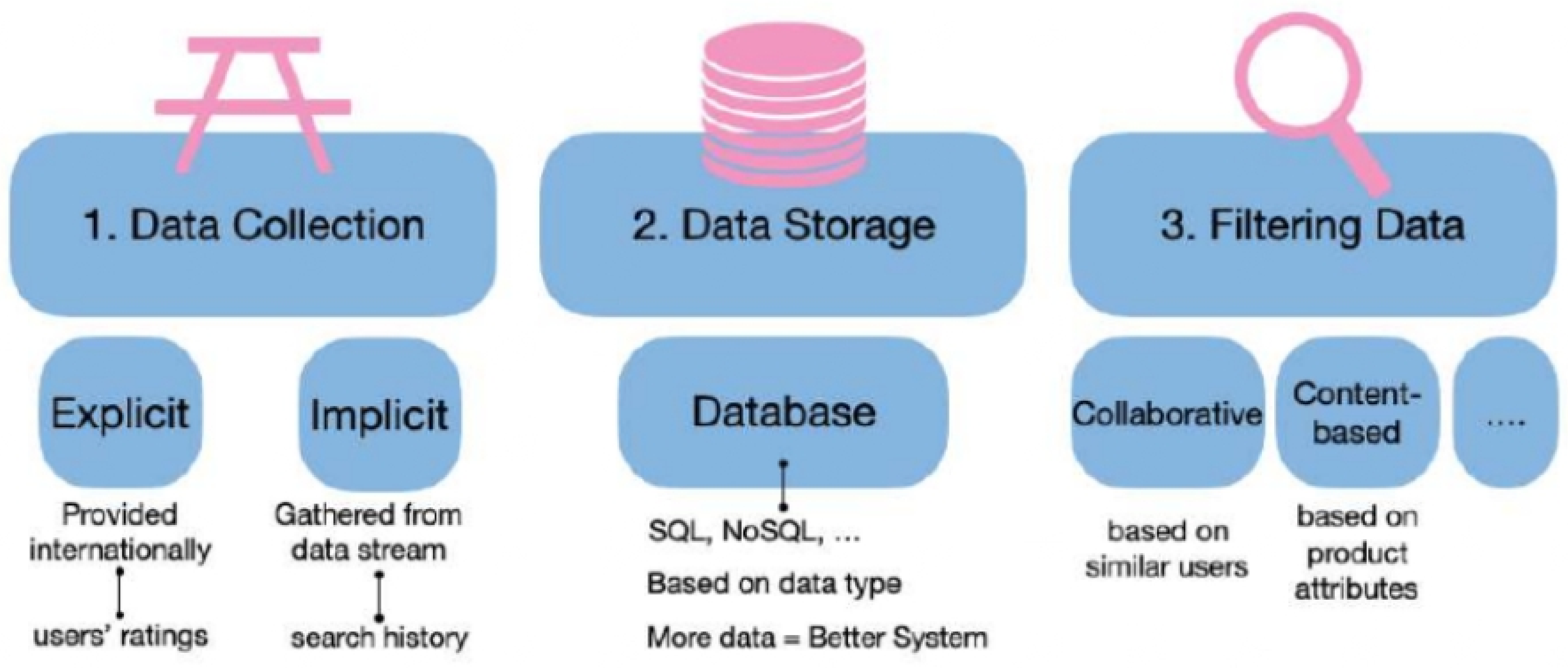
Scikit-Learn is a free machine learning library for Python. It supports both supervised and unsupervised machine learning, providing diverse algorithms for classification, regression, clustering, and dimensionality reduction. It is licensed under a permissive simplified BSD licence and is distributed under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis

-> Recommendation System:
Simply put a Recommendation System is a filtration program whose prime goal is to predict the "rating" or "preference" of a user towards a domain-specific item or item. In our case, this domain-specific item is a movie, therefore the main focus of our recommendation system is to filter and predict only those movies which a user would prefer given some data about the user him or herself.

System Mechanism:



The approach to build the movie recommendation engine consists of the following:
1. Perform Exploratory Data Analysis (EDA) on the data.
2. Build the recommendation system.
3. Get recommendations.

Dataset

By using the movie dataset we are going to recommend the movie based on their ratings, release date and customer interest.
Attributes will be:

Index	(movie no.)
Id	(User Id)
Original Title	(Movie name)
Popularity	(Ratings)
Release Date	(To know whether the movie is latest or old)

Input & Output

Input:

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

csv_url="https://github.com/rashida048/Some-NLP-Projects/blo
b/master/movie_dataset.csv"
col_names=['index', 'id', 'original title', 'popularity',
'release date']
df = pd.read_csv(csv_url, names= col_names)

features = ['id','popularity']

def combine_features(row):
    return row["original title"]+" "+row["release date"]

for feature in features:
    df[feature] = df[feature].fillna('')
df["combined_features"] = df.apply(combine_features,axis=1)
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])
cosine_sim = cosine_similarity(count_matrix)

def get_title_from_index(index):
    return df[df.index == index]["original title"].values[0]

def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]

movie_user_likes = "Avatar"
movie_index = get_index_from_title(movie_user_likes)
similar_movies =  list(enumerate(cosine_sim[movie_index]))
```

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

i=0
print("Top 5 similar movies to "+movie_user_likes+" are:\n")
for element in sorted_similar_movies:
    print(get_title_from_index(element[0]))
    i=i+1
    if i>=5:
        break
```

Output:

```
Top 5 similar movies to Avatar are:

Guardians of the Galaxy

Aliens

Star Wars: Clone Wars: Volume 1

Star Trek Into Darkness

Star Trek Beyond
```

Conclusion

Recommendation systems have become an important part of everyone's lives. With the enormous number of movies releasing worldwide every year, people often miss out on some amazing work of arts due to the lack of correct suggestions. Putting machine learning based Recommendation systems into work is thus very important to get the right recommendations. We saw content-based recommendation systems that although may not seem very effective on its own, but when combined with collaborative techniques can solve the cold start problems that collaborative filtering methods face when run independently.