Movie Recommendation Based On Emotion using Web Scraping with IBM

1.INTRODUCTION

1.1 OVERVIEW

It’s not always easy to pick the right movie to watch. Sometimes you're in the state of mind to see individuals begin to look all starry eyed at, or you need a motion picture to remind your connection with music, or you need to just watch characters to whom you relate or any emotion-based movie. Settling on this decision can't generally be understood by attempting to pick something dependent on genre. This is why we are going to create, a “**Movie recommendation system based on emotion”**, which enables us to choose movies based on how you want their viewing experience to make them feel. An Emotion-based Recommender System (E-MRS) captures customer preferences according to their emotions. Incorporate user emotions into the recommendation process. One of the underlying targets of movies is to evoke emotions in their viewers. IMDb offers all the movies for all genres. Therefore the movie titles can be scraped from the IMDb list to recommend to the user. IMDb does not have an API, for accessing the information on movies and TV Series. Therefore we have to perform scraping. Scraping is used for accessing information from a website which is usually done with APIs.The scraper is written in Python and uses lxml for parsing the web pages. BeautifulSoup is used for pulling data out of HTML and XML files. Anticipation’, ‘Fear’, ‘Joy’, ‘Sad’, ‘Trust’. Here these are taken as input and the corresponding movies would be displayed for the emotion. The correspondence of every emotion with the genre of movies is listed below: Based on the input emotion, the corresponding genre would be selected and all the top movies of that genre would be recommended to the user.

1.2 **PURPOSE**

An Emotion-based Recommender System (E-MRS) captures customer preferences according to their emotions. Emotion plays an important role in rational and intelligent behavior, thus, we incorporate user emotions into the recommendation process. One of the underlying targets of movies is to evoke emotions in their viewers. IMDb offers all the movies for all genres. Therefore the movie titles will be scraped from the IMDb list to recommend to the user. IMDb does not have an API, for accessing the information on movies and TV Series. Therefore we have to perform scraping. Scraping is used for accessing information from a website which is usually done with APIs.The scraper is written in Python and uses lxml for parsing the web pages. BeautifulSoup is used for pulling data out of HTML and XML files. Anticipation', 'Fear', 'Joy', 'Sad', 'Trust'. Here these are taken as input and the corresponding movies would be displayed for the emotion. The correspondence of every emotion with the genre of movies is listed below: Based on the input emotion, the corresponding genre would be selected and all the top movies of that genre would be recommended to the user.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

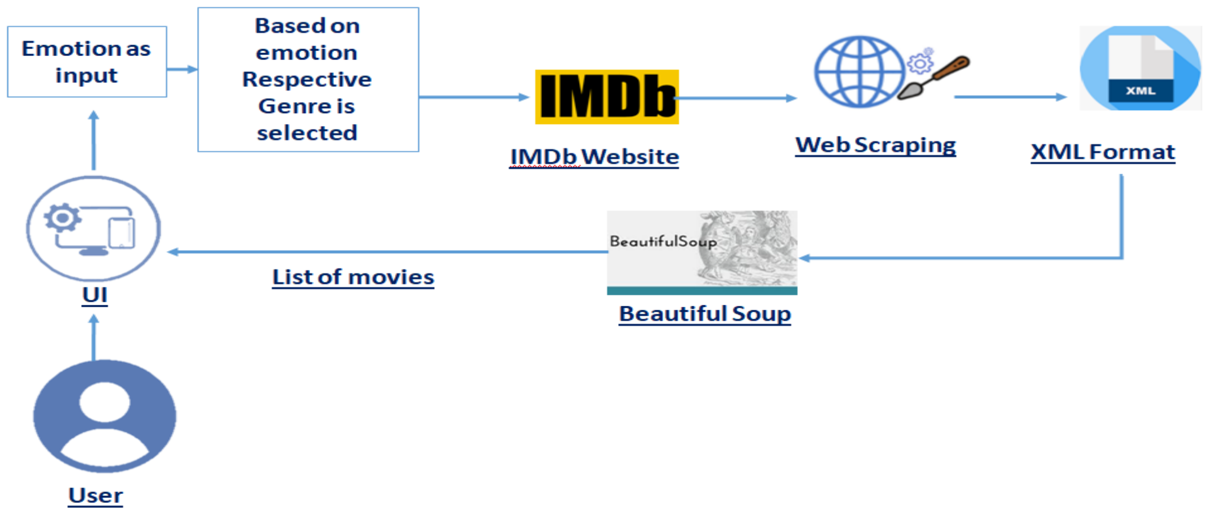
The customer usually provides the recommender system with data such as the characteristics of the product he is looking for, his ratings, demographic data, etc. The recommender system applies one or several recommendation techniques to these data and then recommends products to the customers. In order to provide reliable recommendations, the recommender system needs to capture exactly the customer's needs and preferences. However, for subjective and complex products such as movies, music, perfume, the task of rating or describing the desired product characteristics is quite difficult for customers. Moreover, as user preferences for these subjective products change constantly according to their emotions, the traditional user profile is not sufficient to understand and capture these changes.Large number of unsorted movies are available online. Viewer is unable to find the movies that he/she wants to watch. Movies are not available on the basis of mood/emotion of the user. Existing software doesn't suggest movies to improve viewer's mood.

2.2 **PROPOSED SYSTEM**

The main purpose of the Emotion-based Recommender System (E-MRS) is to suggest a list of movies to the viewer, from the large amount of content available online, based on emoji. We are making this software that is based on the present mood of the user.This software will give movie suggestion to the emoji.This software will suggest them latest and trending movies on their mood from a large variety of movies online by "Web Scraping". Another purpose is to suggest list of latest and trending movies.Emojis will be given as input and the corresponding movies would be displayed for the emoji. The correspondence of every emotion with the genre of movies will be listed. Based on the input emotion, the corresponding genre would be selected and all the top movies of that genre would be recommended to the user.

3. THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE AND SOFTWARE DESIGNING

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was created by Guido van Rossum, and first released on February 20, 1991. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Anaconda Navigator

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder

Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Spyder

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing,debugging, and introspection features. Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.Spyder is extensible with first-party and third party plugins includes support for interactive tools for data inspection and embeds Pythonspecific code. Spyder is also pre-installed in Anaconda Navigator, which is included in Anaconda.

Tensor flow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML-powered applications.

Keras

Keras leverages various optimization techniques to make high-level neural network API easier and more performant. It supports the following features:

- Consistent, simple, and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is a user-friendly framework that runs on both CPU and GPU.
- Highly scalability of computation.

Flask

Web framework used for building. It is a web application framework written in python which will be running in local browser with a user interface. In this application, whenever the user interacts with UI and selects emoji, it will suggest the best and top movies of that genre to the user.

Web Scraping using Python

Web Scraping refers to access the HTML of the webpage and extract useful information and data from it. This method is also called web scratching or web reaping or web information extraction. Here in this task, we are going to utilize web scratching to separate information from the website page utilizing Python and BeautifulSoup**.** The scraper is written in Python and utilizations lxml for parsing the site pages.

LXML

Python lxml is the most feature-rich and simple to-utilize library for processing XML and HTML data. Python contents are composed to perform numerous errands like Web scraping or scratching and parsing XML.

BeautifulSoup

It is a library of python which is utilized to pull the data from the web pages i.e HTML and XML files. It works with your preferred parser to give colloquial methods for exploring, looking and changing the parse tree.

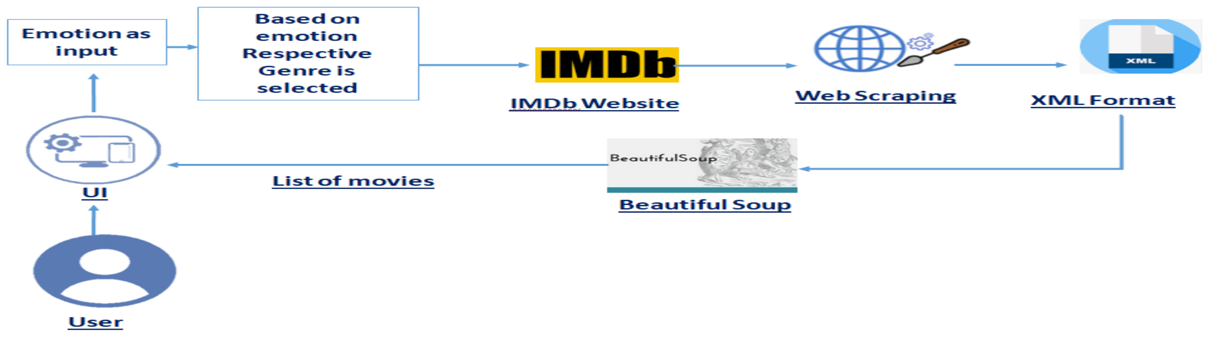
Hardware Requirements:

o Operating system: window 7 and above with 64bit
o Processor Type -Intel Core i3-3220
o RAM: 4Gb and above
o Hard disk: min 100GB

4. EXPERIMENTAL INVESTIGATION

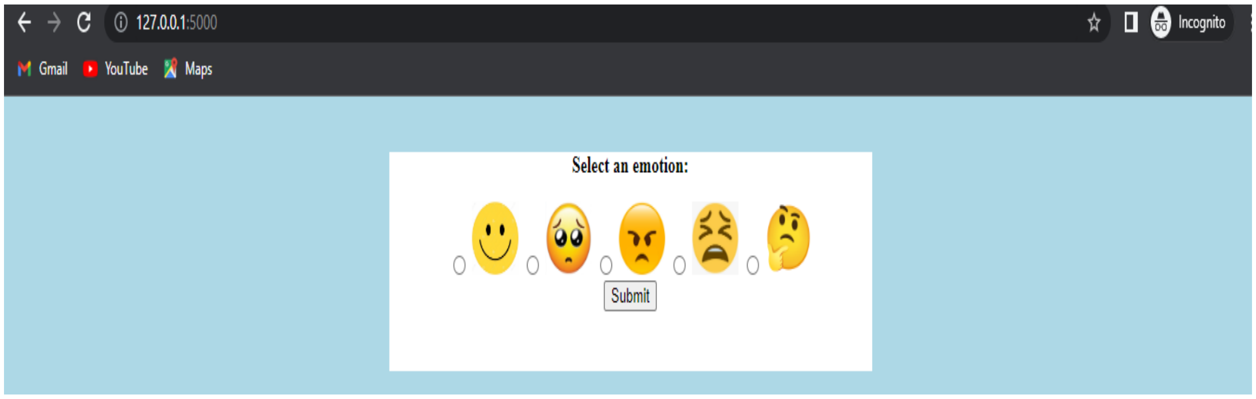
Up on several investigations it is found that customers are happy to use the Emotion based movie recommender system as it recommend a list of movies to the viewer, from the large amount of content available online, based on emoji. Also in future the scope of the model can be extended beyond imagination. Based on the above investigation this idea is useful to put into use.

5. FLOWCHART



6.RESULT

1.

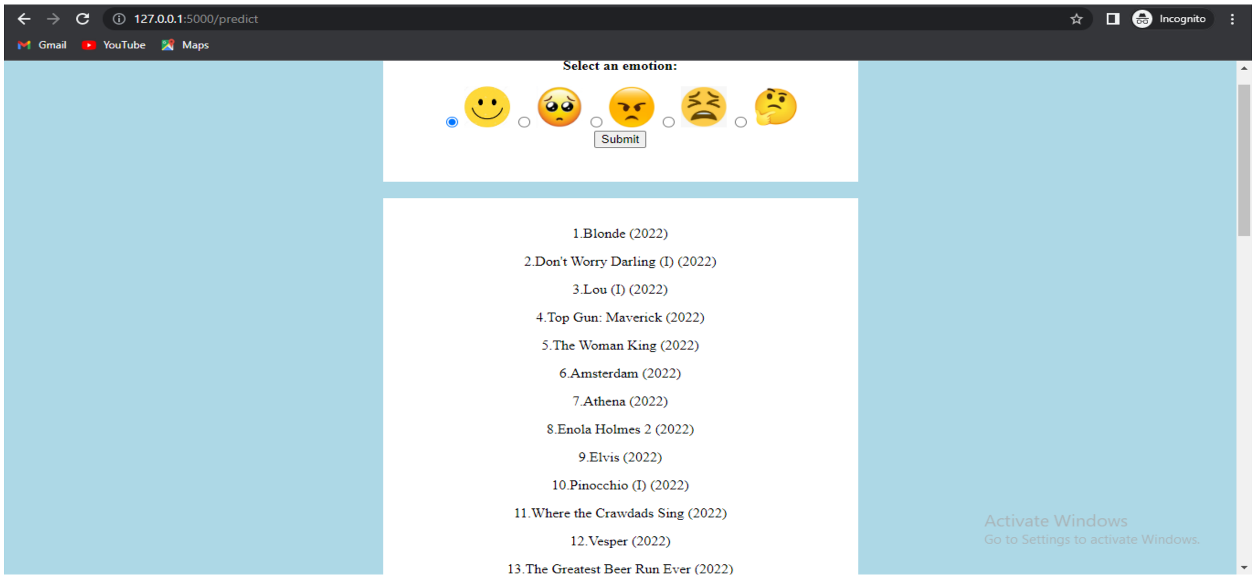


127.0.0.1:5000

Select an emotion:

Submit

2.



127.0.0.1:5000/predict

Select an emotion:

Submit

1.Blonde (2022)

2.Don't Worry Darling (I) (2022)

3.Lou (I) (2022)

4.Top Gun: Maverick (2022)

5.The Woman King (2022)

6.Amsterdam (2022)

7.Athena (2022)

8.Enola Holmes 2 (2022)

9.Elvis (2022)

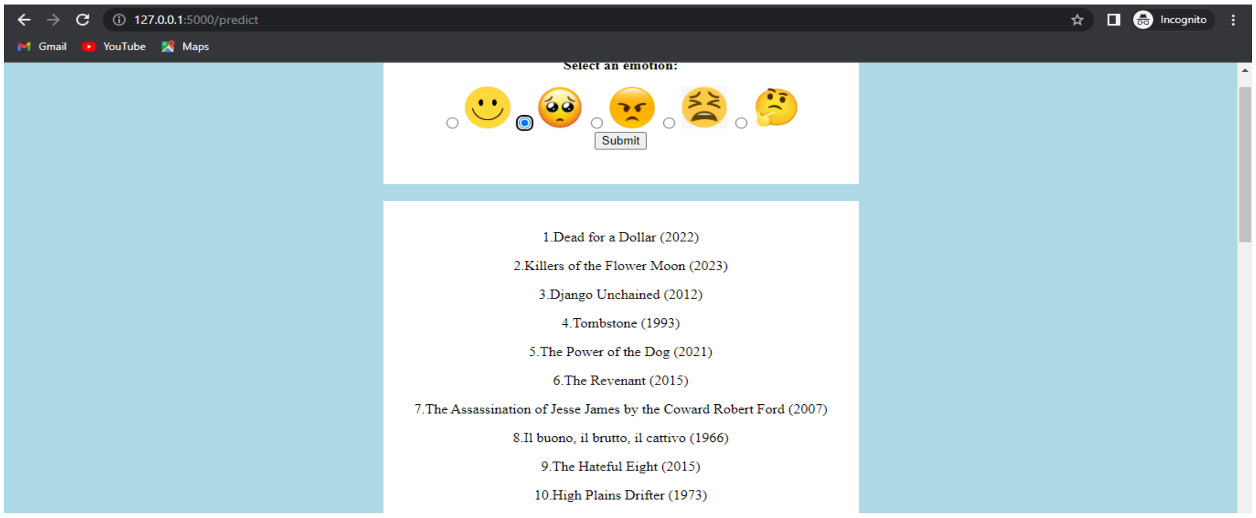
10.Pinocchio (I) (2022)

11.Where the Crawdads Sing (2022)

12.Vesper (2022)

13.The Greatest Beer Run Ever (2022)

3.



127.0.0.1:5000/predict

Select an emotion:

Submit

1.Dead for a Dollar (2022)

2.Killers of the Flower Moon (2023)

3.Django Unchained (2012)

4.Tombstone (1993)

5.The Power of the Dog (2021)

6.The Revenant (2015)

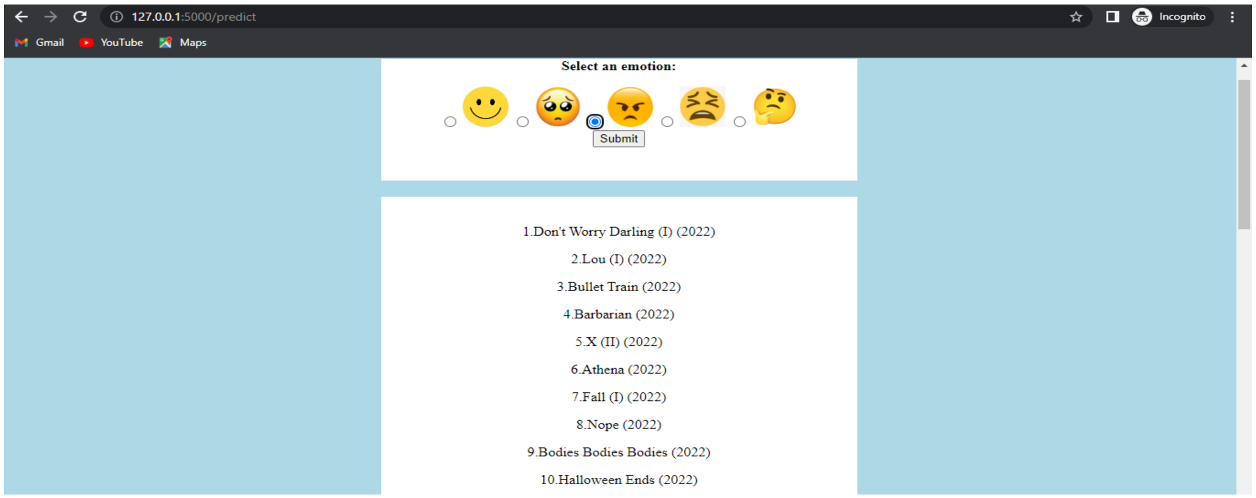
7.The Assassination of Jesse James by the Coward Robert Ford (2007)

8.Il buono, il brutto, il cattivo (1966)

9.The Hateful Eight (2015)

10.High Plains Drifter (1973)

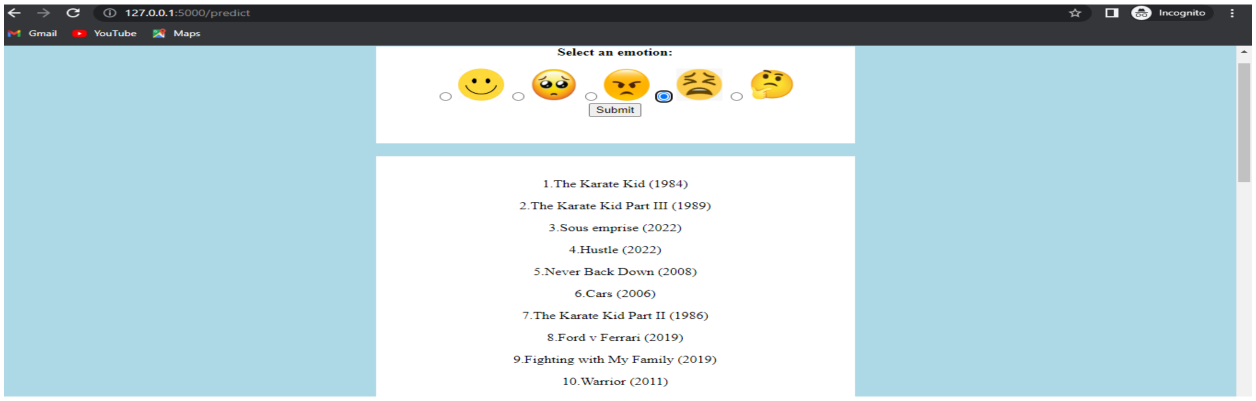
4.



Select an emotion:

1.Don't Worry Darling (I) (2022)
2.Lou (I) (2022)
3.Bullet Train (2022)
4.Barbarian (2022)
5.X (II) (2022)
6.Athena (2022)
7.Fall (I) (2022)
8.Nope (2022)
9.Bodies Bodies Bodies (2022)
10.Halloween Ends (2022)

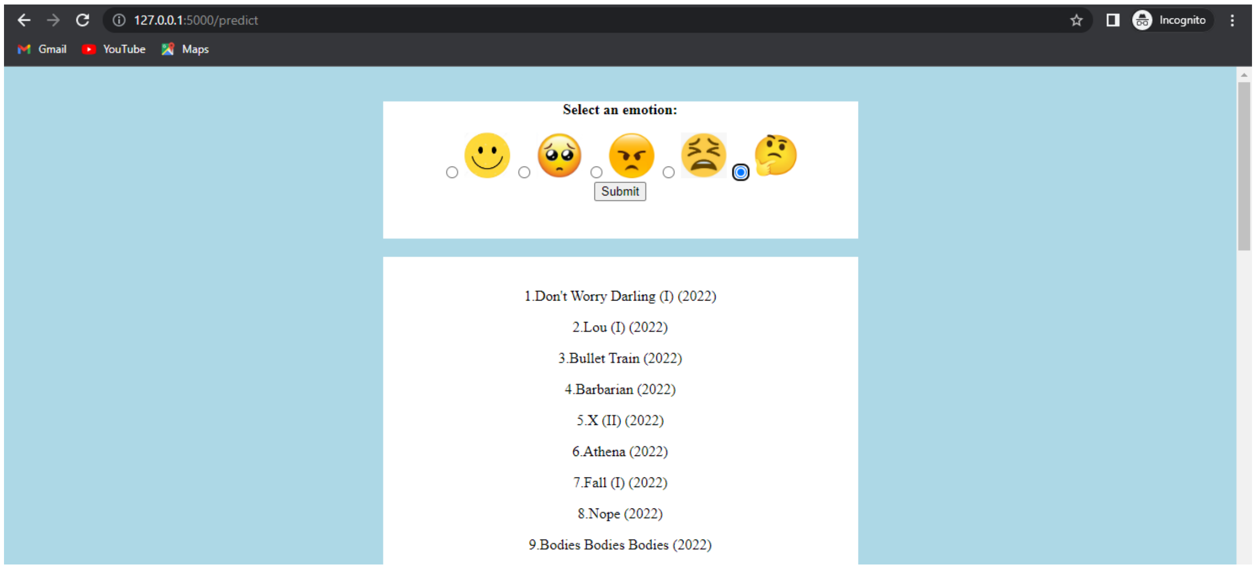
5.



Select an emotion:

1.The Karate Kid (1984)
2.The Karate Kid Part III (1989)
3.Sous emprise (2022)
4.Hustle (2022)
5.Never Back Down (2008)
6.Cars (2006)
7.The Karate Kid Part II (1986)
8.Ford v Ferrari (2019)
9.Fighting with My Family (2019)
10.Warrior (2011)

6.



Select an emotion:

1.Don't Worry Darling (I) (2022)
2.Lou (I) (2022)
3.Bullet Train (2022)
4.Barbarian (2022)
5.X (II) (2022)
6.Athena (2022)
7.Fall (I) (2022)
8.Nope (2022)
9.Bodies Bodies Bodies (2022)

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGE

Customer get what they wants , there is no need to search for the movies, customer satisfactions remains high, leading to high customer retention. Provides recommendation makes users feel free brands really care about their needs and wants,hence keeping them engaged and loyal.

DISADVANTAGE

If the system recomends products with bias, then customer wil be landing into wrong deals. Chances are that some websites may suggest products wrongly based on analysis of little information gathered.

8. APPLICATION

Work on several numbers of data: The number of choices for anything on internet is veryhigh and it's tedious to refine most wanted data by self while searching. The scope of this proposal system includes working within numerous data, with case.

Saving of time: Many people have problem selecting the alternative item of movie due to lack of time and due to search issues. Also movie recommendations from friends can be time consuming. The system helps in saving lots of time.

9. CONCLUSION

This project has the novelty of incorporating user emotions into the user profile to provide users with well recommended products based on their emotional state. Because movie is a complex and subjective domain, it is necessary to incorporate user emotion into the user profile. Users can give their feedback about how a recommended movie meets their preferences. This feedback (in the form of user’s rating) improves the recommendation quality over time. Because emotion can influence interactions, behaviors and thinking of the user, we believe that E-MRS with the Emotion detector can greatly improve the efficiency of the movie recommendation.

10. FUTURE SCOPE

The user interface through the website can be made easier to access through emoji detection Data regarding the movies for which the recommendations are requested can also be gathered from several websites using web scraping or web data extraction tools after obtaining legal permissions. This can also be developed as a standalone application (engine) which can be used by small e-commerce site vendors to acquire and attach to their sites. For processing large scale data, the application can be integrated along with intelligent data analyses using big data techniques to provide authentic and accurate analytics. Sentiment analysis can also be applied to "comments" information to identify the emotion behind the comments (positive, negative or neutral) to recommend movies appropriately.

11. BIBILOGRAPHY

https://www.tutorialspoint.com/flask/flask_application.htm
https://python.plainenglish.io/how-to-scrape-imdb-data-9d7535b98576
https://www.w3resource.com/python-exercises/web-scraping/web-scraping-exercise-23.php
https://www.geeksforgeeks.org/recommendation-system-in-python/

APPENDIX

app.py

```
from flask import Flask, request, render_template
import numpy as np
import re
import os
from gevent.pywsgi import WSGIServer
import requests as HTTP
from bs4 import BeautifulSoup as SOUP
app = Flask(__name__)
app=Flask(__name__,template_folder="templates")
@app.route('/',methods=['GET'])
def index():
return render_template('home.html')
```

```
@app.route('/home', methods=['GET'])
def about():
 return render_template('home.html')
@app.route('/predict', methods=["GET","POST"])
def predict():
    #emotion=None

#urlhere='http://www.imdb.com/search/title?genres=drama&title_type=feature&sort=m
oviemeter, asc'
 if request.method == "POST":
      emotion=request.form['emotion']
    print(emotion)
    if(emotion == "happy"):
                                                                 urlhere        =
'http://www.imdb.com/search/title?genres=drama&title_type=feature&sort=moviemeter,
asc'
    elif(emotion == "angry"):
                                                                 urlhere        =
'http://www.imdb.com/search/title?genres=thriller&title_type=feature&sort=moviemeter,
asc'
    elif(emotion == "disgust"):
                                                                 urlhere        =
'http://www.imdb.com/search/title?genres=sport&title_type=feature&sort=moviemeter,
asc'
    elif(emotion == "think"):
                                                                 urlhere        =
'http://www.imdb.com/search/title?genres=thriller&title_type=feature&sort=moviemeter,
asc'
 elif(emotion == "sad"):
                                                                 urlhere        =
'http://www.imdb.com/search/title?genres=western&title_type=feature&sort=moviemet
er, asc'
```

```
 response = HTTP.get(urlhere)
  data = response.text
  soup = SOUP(data, "lxml")
  supa = soup.find_all('h3', attrs={'class' : 'lister-item-header'})
  list = []
  for header in supa:
        name = "";
        aElement_soup = header.find_all('a')
        spanElement_soup = header.find_all('span')
        spanElement = spanElement_soup[0]
        name = name + spanElement.text
        aElement = aElement_soup[0]
        name = name + "" + aElement.text
        if len(spanElement_soup)>1:
         spanElement = spanElement_soup[1]
         name = name + "\n" + spanElement.text
        list.append(name)

   return render_template('home.html',prediction_text="{}".format(emotion),data=list)
  if __name__ == "__main__":
  app.run(debug=False)
```

home.html

```
<html>
<head>
<style>
.myDiv {
  position:relative;
  top: 30;
  width: 520;
  height: 150;
  margin: auto;
  background-color: white;
  text-align: center;
  align: center;
}
body {
background-color:lightblue;
}
.bDiv{
    position: relative;
    top:50;
    width:520;
    height: 1800;
    margin: auto;
    background-color: white;
    text-align: center;
    align: center;
  }
</style>
</head>
<body>
<div class="myDiv" align="center" >
<p><b>Select an emotion:</b></p>
<form action="/predict" method="POST"  >
  <input type="radio" id="html" name="emotion" value="happy">
  <label for="happy"><img src="static/img/happy.jpg" width="50" height="50"></label>
 <input type="radio" id="html" name="emotion" value="sad">
  <label for="sad"><img src="static/img/sad.jpg" width="50" height="50"></label>
<input type="radio" id="html" name="emotion" value="angry">
  <label for="angry"><img src="static/img/angry.jpg" width="50" height="50"></label>
 <input type="radio" id="html" name="emotion" value="disgust">
  <label for="disgust"><img src="static/img/disgust.png" width="50"
height="50"></label>
```

```
  <input type="radio" id="html" name="emotion" value="think">
  <label for="think"><img src="static/img/think.png" width="50" height="50"></label>
  <br>
 <input type="submit" value="Submit">
</form>
</div>
<div class="bDiv">
<br>
{% for data_element in data %}
 <p>{{data_element}}</p>
{% endfor %}
</div>
</body>
</html>
```