NETFLIX RECOMMENDATION SYSTEM

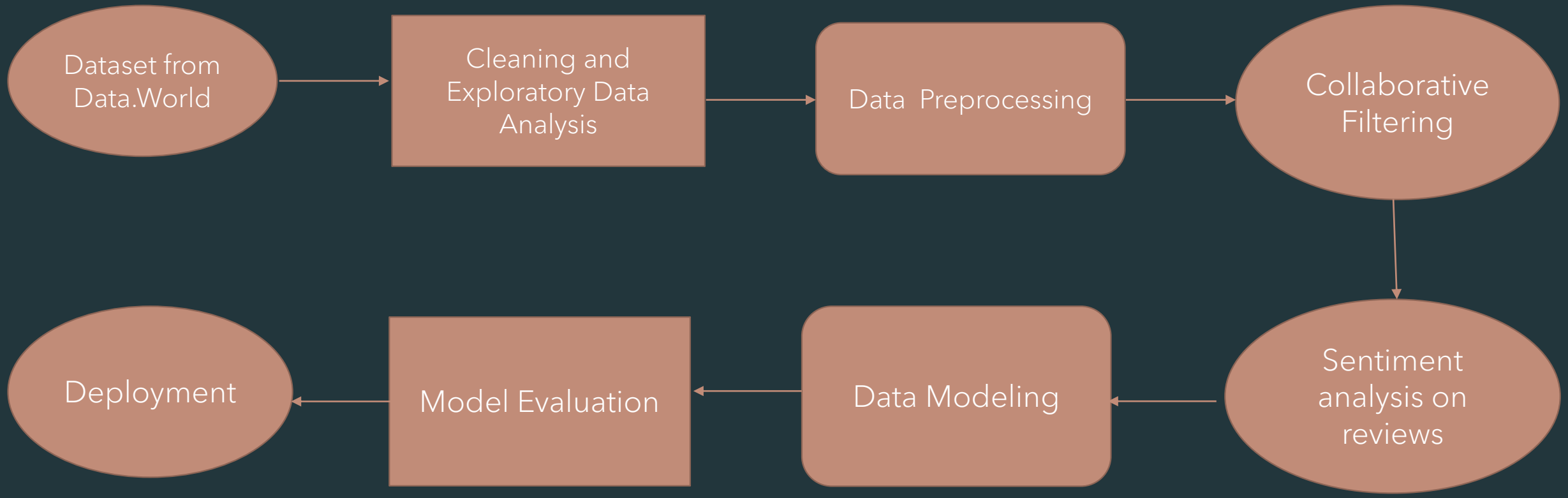
Capstone Project

By :Mercy Subramani

Semester : Spring2023

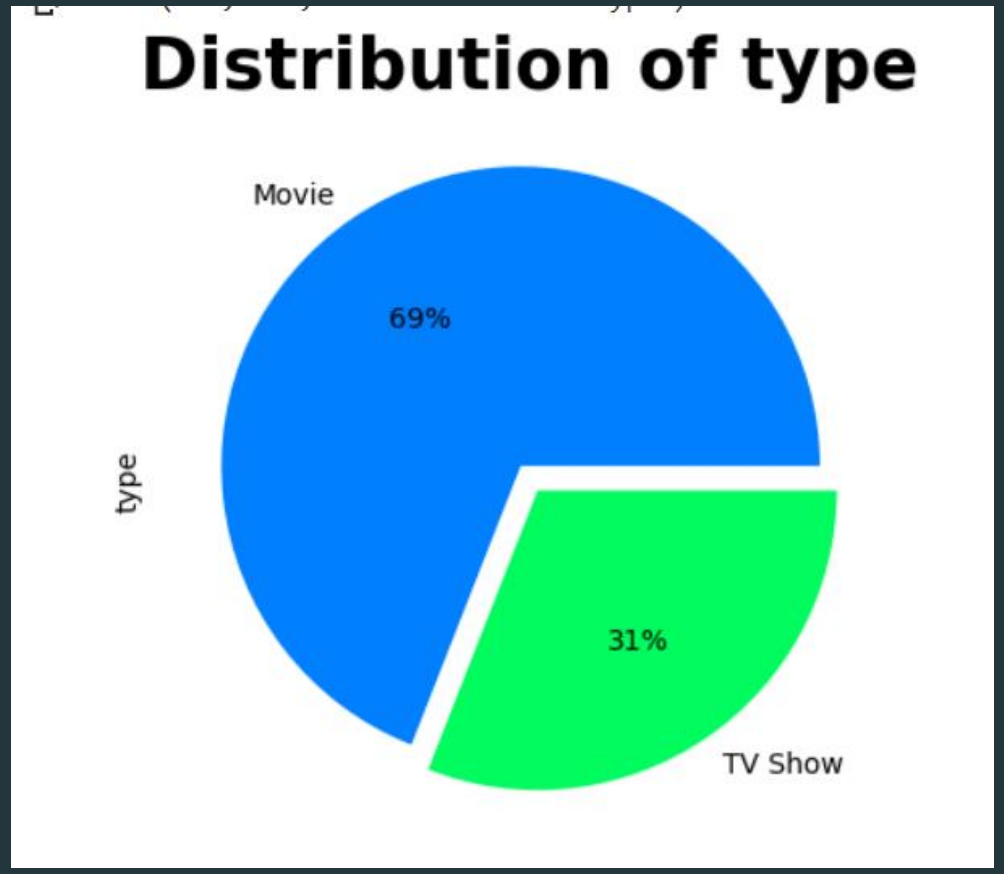
*Instructor : Dr. **Chaojie Wang***

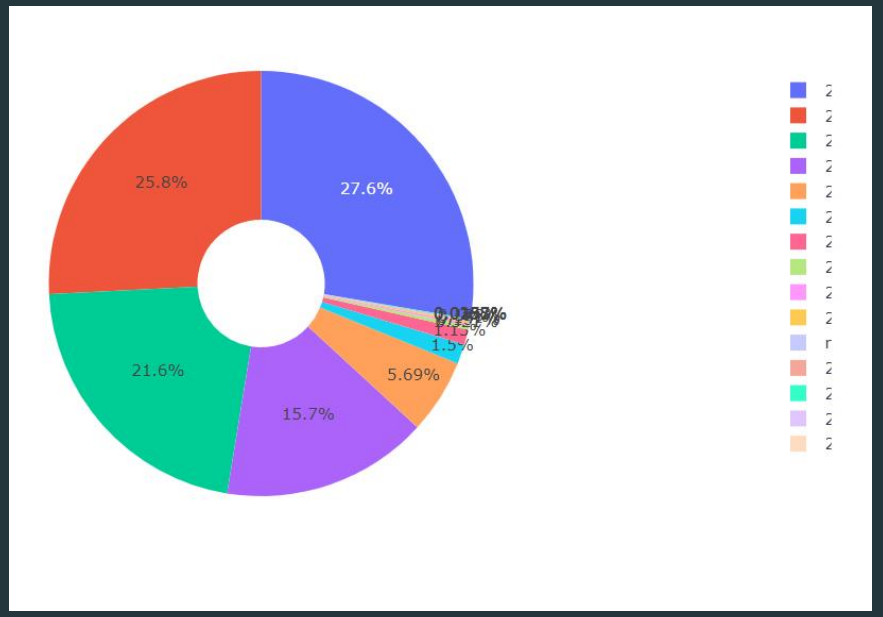
PROJECT PROCESS FLOW DIAGRAM



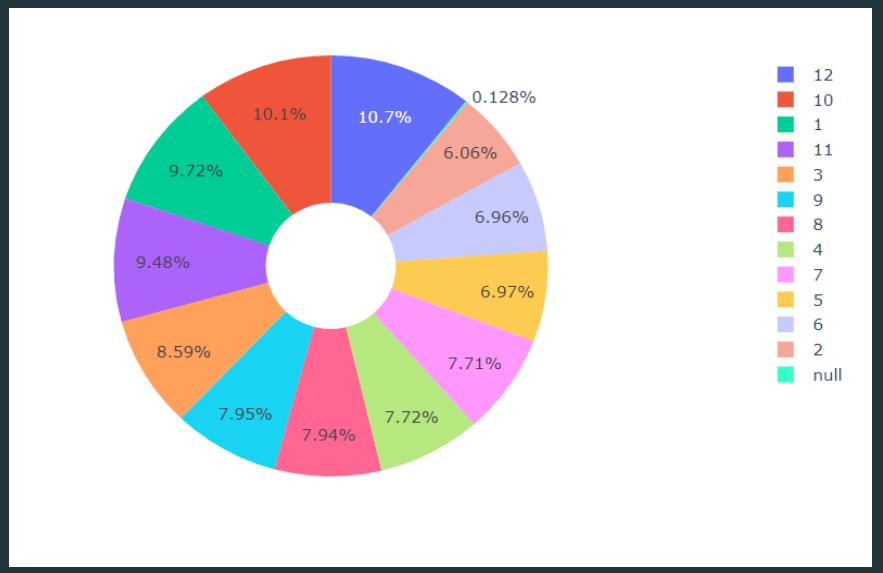
EXPLORATORY DATA ANALYSIS

Movie vs Tv show





Yearly Distribution

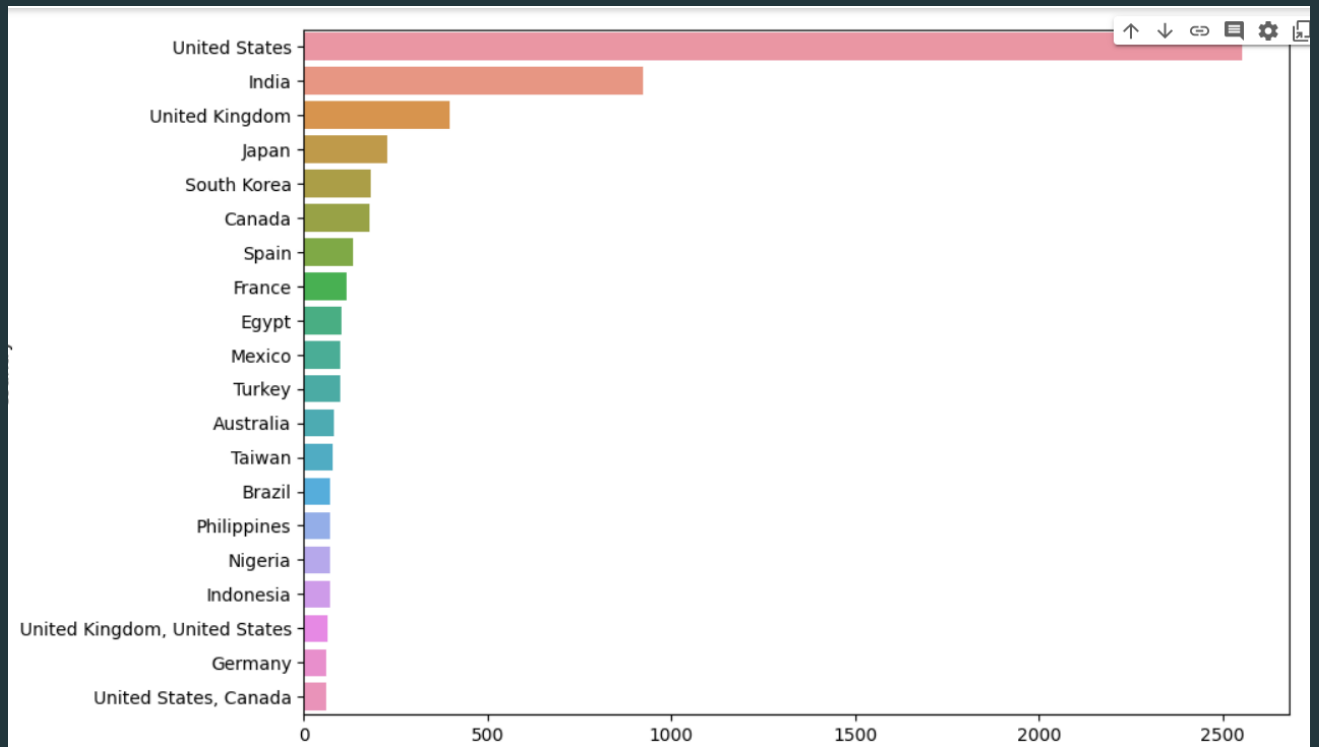


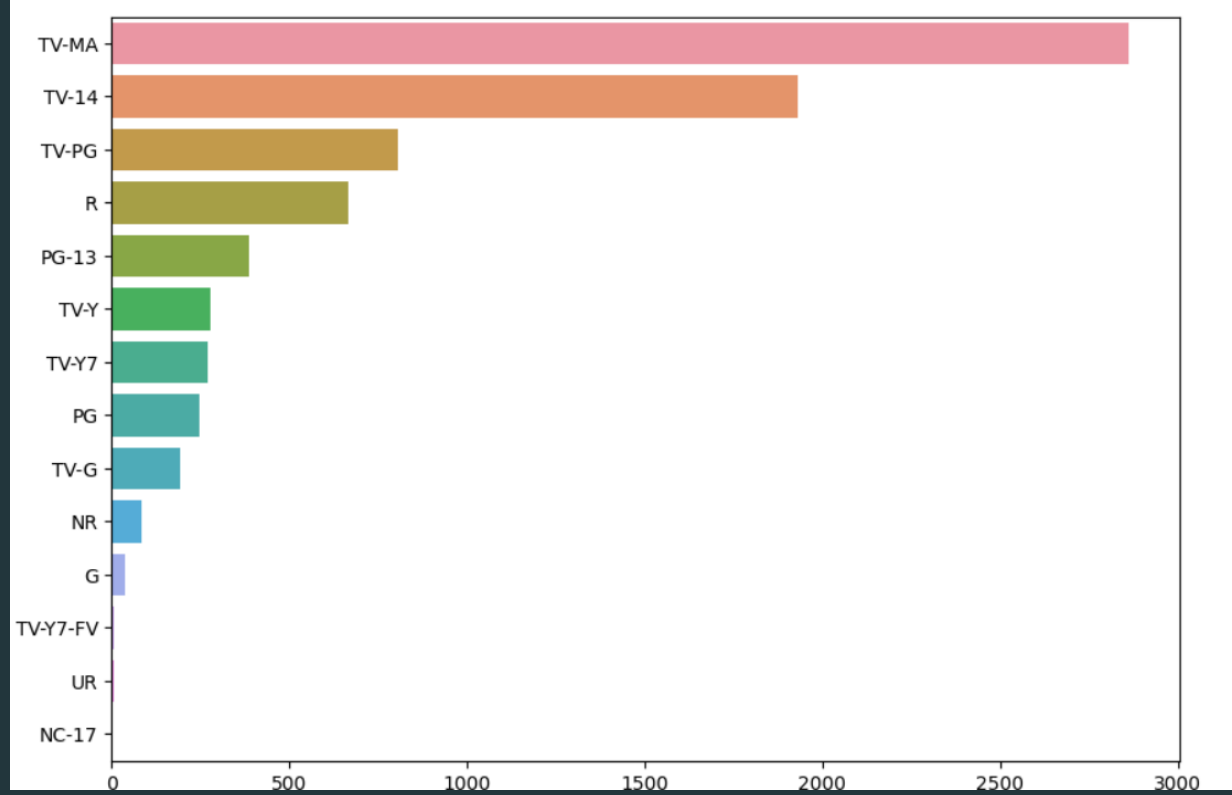
Monthly Distribution

EDA CONTINUATION

Visualization shows that United states has more movies and tv shows than other countries

Next comes India and United Kingdom which have more movies in Netflix





Graph shows that Adult movies and Tv shows are more

DATA PREPROCESSING

Stemming

- Used reduce words to its base
- For example: “running” – “runs”
- Application- search engines, text classification, Sentiment analysis

Count Vectorization

- Text preprocessing technique
- Convert collection of text documents into Numerical representation

```
[ ]  from nltk.corpus import stopwords
     from nltk.stem.snowball import SnowballStemmer

[ ]  stemmer = SnowballStemmer("english")

     def Apply_stemming(text):
       text = [stemmer.stem(word) for word in text.split()]
       return " ".join(text)

[ ]  df['description'] = df['description'].apply(Apply_stemming)
     df.head()
```

COSINE SIMILARITY

- Measures similarity between two non-zero vectors
- Here, cosine similarity is used to calculate items or user based feature vectors
- recommends similar items
- It is dot product-based algorithm

```
def recommend(movie):
  movie_index = df[df['title']==movie].index[0]
  distance = similarity[movie_index]
  movie_list = sorted(list(enumerate(distance)),reverse=True, key=lambda x:x[1])[1:6]

  for i in movie_list:
    print(df.iloc[i[0]].title)
```

```
[ ] recommend("3 Heroines")
```

```
Sparring
Dangal
Azhar
Ladies First
Summer of '92
```

WEB SCRAPING

- Web scraping was done by beautifulsoup
- For example, I have taken avengers end game Reviews
- And used nlp based libraries like stopwords, punctuation to clean data

```
import requests
from bs4 import BeautifulSoup
```

```
[ ] movie_reviews = []
    url = 'https://www.google.com/search?q=movie+reviews+avengers+endgame
```

```
[ ] response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    for review in soup.find_all('div', {'class': 'b1hJbf'}):
        movie_reviews.append(review.get_text())
```

```
[ ] import nltk
    from nltk.corpus import stopwords
    from string import punctuation

    # Download stopwords and punkt tokenizer
    nltk.download('stopwords')
    nltk.download('punkt')

    # Load stop words and punctuations
    stop_words = set(stopwords.words('english'))
    punctuations = set(punctuation)
```

SENTIMENT ANALYSIS

- Sentiment Analysis was performed on reviews
- It will help to recommend the similar movie with similar sentiments
- -1 is Negative
- +1 is Positive

```
] labels =['positive', 'negative', 'positive', 'positive', 'negative', 'positive', 'negative', 'negative', 'positive', 'positive']

] df = pd.DataFrame({'review':text, 'sentiment': labels})

  X_train, X_test, y_train, y_test = train_test_split(df['review'], df['sentiment'], test_size=0.2, random_state=42)

] vectorizer = TfidfVectorizer()
  X_train_vec = vectorizer.fit_transform(X_train)
  X_test_vec = vectorizer.transform(X_test)
```

DATA MODELING: NAÏVE BAYES

- Naïve bayes was used on test and train data
- It is a probabilistic classification algorithm
- Used for classification task , sentiment analysis and spam filtering
- It makes the assumption that all features are independent of each other

```
[ ] nb = MultinomialNB()
    nb.fit(X_train_vec, y_train)

    MultinomialNB
    MultinomialNB()

[ ] y_pred = nb.predict(X_test_vec)
    accuracy = accuracy_score(y_test, y_pred)
    print('Accuracy:', accuracy)

    Accuracy: 0.5

[ ] from sklearn.model_selection import cross_val_score
    from sklearn.pipeline import Pipeline
    from sklearn.feature_extraction.text import CountVectorizer

[ ] pipeline = Pipeline([
        ('vectorizer', CountVectorizer(lowercase=True, stop_words='english', strip_accents='unicode')),
        ('classifier', MultinomialNB())
    ])

[ ] scores = cross_val_score(pipeline, df['review'], df['sentiment'], cv=3)

[ ] print(scores)

    [0.5        0.66666667 0.66666667]
```

K NEAREST NEIGHBORS

- K Nearest Neighbors is also used to check the accuracy
- It is a clustering algorithm
- It is a hyperparameter that can adjust based on problems

knn

```
[ ]  from sklearn.neighbors import KNeighborsClassifier
```

```
[ ]  knn = KNeighborsClassifier(n_neighbors=5)
```

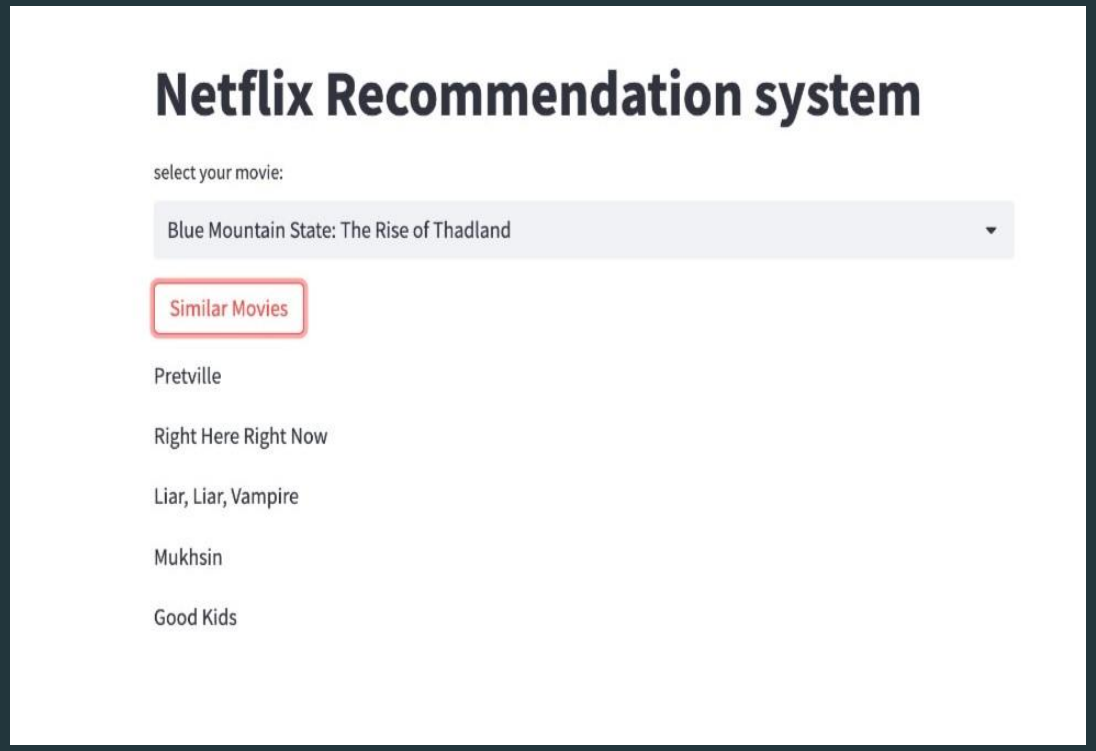
```
[ ]  X_train, X_test, y_train, y_test = train_test_split(df['review'], df['sentiment'], test_size=0.2, random_state=42)
```

```
[ ]  knn.fit(X_train_vec, y_train)

     # Predict the sentiment of the test data
     y_pred = knn.predict(X_test_vec)

     # Calculate the accuracy of the model
     accuracy = accuracy_score(y_test, y_pred)
     print('Accuracy:', accuracy)
```

```
Accuracy: 0.5
```

DEPLOYMENT

- Deployed using Streamlit
- User friendly interface
- No HTML , CSS is needed
- Based on python language




```
[ ]  %%writefile file_py.ipynb
     import streamlit as st

     import pickle
     import pandas as pd

     def recommend(movie):
       print(df)
       movie_index = new_df[new_df['title']==movie].index[0]
       distance = similarity[movie_index]
       movie_list = sorted(list(enumerate(distance)),reverse=True, key=lambda x:x[1])[1:6]
       recommended_movies = []
       for i in movie_list:
         recommended_movies.append(new_df.iloc[i[0]].title)
       return recommended_movies

     df_dict = pickle.load(open('df_dict.pkl','rb'))
     tags = pd.DataFrame(df_dict)

     similarity = pickle.load(open('similarity.pkl','rb'))

     st.title('Movie Recommendation system')

     option= st.selectbox('select your movie:', tags['title'].values)

     if st.button('Recommend'):
       recommendations = recommend(option)
       for i in recommendations:
         st.write(i)###
```


Thank you