Recommended for you and by you

By Group 1 (Beatrice Apikos-Bennett, Andrew Prozorovsky, Shaji Qurashi, Patrick Moon)

Do you struggle with picking that next movie?

What is item-based collaborative filtering?

> “Item based collaborative filtering was introduced 1998 by Amazon[6]. Unlike user based collaborative filtering, **item based filtering** looks at the **similarity between different items**, and does this by taking note of **how many users that bought item X also bought item Y**. If the correlation is high enough, a similarity can be presumed to exist between the two items, and they can be assumed to be similar to one another.”

- *Comparison of User Based and Item Based Collaborative Filtering Recommendation Services*, BOSTRÖM and FILIPSSON

Steps to build our *recommender*

1 | Find data

MovieLens is a wonderful and large dataset available for academic use. We are thankful they gave us permission to utilize their data!

2 | Clean data

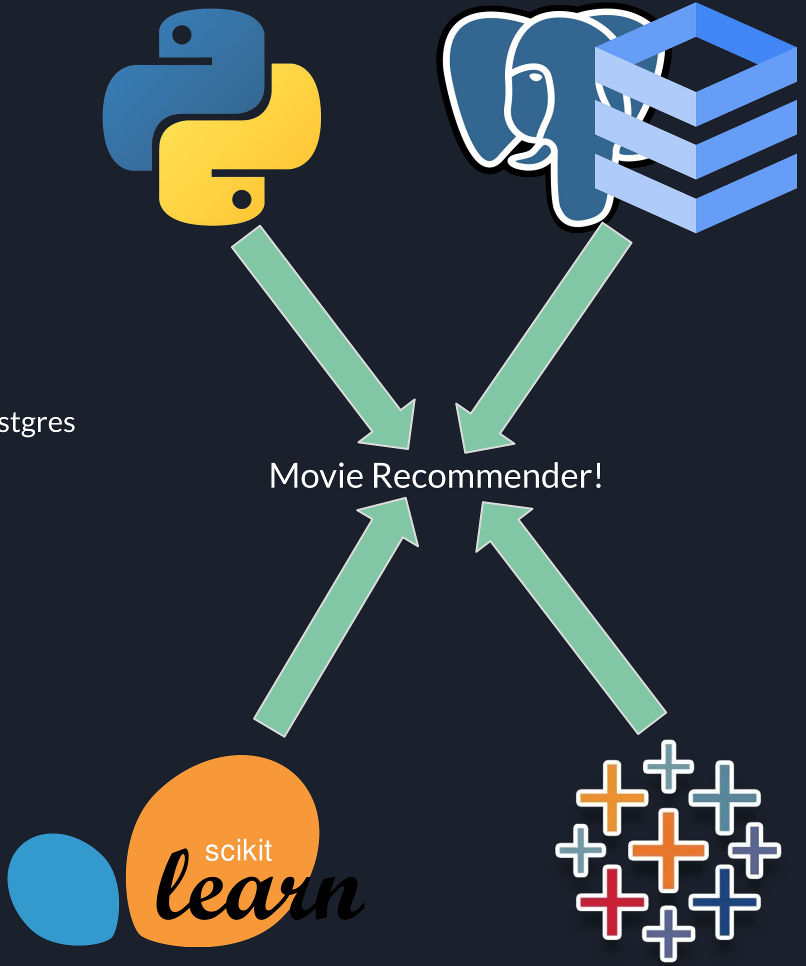
- CSVs fed into Postgres
- Separated year from title and reformatted
- Isolated genre instances
- Removed duplicates
- Bulk formatted data in Postgres using DML
- Queried data from Postgres to include necessary data and columns based on our needs
- Assigned fake names to users (for fun)
- Etc…

3 | Build out

Utilized k-nearest neighbors to find movies similar to the entry movie by rating across users and return 10 recommendations sorted by their similarity score

Technology used

- Postgres for storing the MovieLens data
- Google Cloud SQL for hosting Postgres instance
  - Allowed collaboration and multiple users to access the same Postgres database
- SQLAlchemy to create a live connection to our cloud-hosted database and create charts
- Pandas and NumPy for model input data manipulation
- Scikit-Learn for building the predictive model
  - NearestNeighbors
  - KNeighborsRegressor
  - Mean Squared Error (MSE) for accuracy testing
- Tableau for visualizing metrics of our source data
- HTML and CSS for building a website



Source file

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy

Data Transformation in Postgres

project4_db/postgres@project4_server

No limit

Query   Query History

```sql
select x.movieid,
x.title,
x.released_year,
mov.title || ' (' || mov.released_year || ')' as "title_released_year",
x.genres
from movies mov
cross join lateral (values
    (mov.movieid, mov.title, mov.released_year, mov.genre_1),
    (mov.movieid, mov.title, mov.released_year, mov.genre_2),
    (mov.movieid, mov.title, mov.released_year, mov.genre_3),
    (mov.movieid, mov.title, mov.released_year, mov.genre_4),
    (mov.movieid, mov.title, mov.released_year, mov.genre_5),
    (mov.movieid, mov.title, mov.released_year, mov.genre_6),
    (mov.movieid, mov.title, mov.released_year, mov.genre_7),
    (mov.movieid, mov.title, mov.released_year, mov.genre_8),
    (mov.movieid, mov.title, mov.released_year, mov.genre_9),
    (mov.movieid, mov.title, mov.released_year, mov.genre_10)
) as x(movieid, title, released_year, genres)
where x.genres is not null
order by mov.title, mov.movieid, x.genres
```

Output data to use for predictive model and Tableau

movieid integer	title character varying (1024)	released_year integer	title_released_year text	genres character varying (255)
1	Toy Story	1995	Toy Story (1995)	Adventure
1	Toy Story	1995	Toy Story (1995)	Animation
1	Toy Story	1995	Toy Story (1995)	Children
1	Toy Story	1995	Toy Story (1995)	Comedy
1	Toy Story	1995	Toy Story (1995)	Fantasy
2	Jumanji	1995	Jumanji (1995)	Adventure
2	Jumanji	1995	Jumanji (1995)	Children
2	Jumanji	1995	Jumanji (1995)	Fantasy

Pros

1. **Large and small data = win**
   Works with various sized data sets
2. **Climate change, fishing, and pollution affect the reef**
   Lorem ipsum dolor sit amet, consectetur adipiscing elit
3. **This area occupies 132,973 mi^2**
   Lorem ipsum dolor sit amet, consectetur adipiscing elit

Cons

1. **Requires users to rate movies rather robustly**
   Without ratings it won't know how to recommend this is where user-based collaborative filtering would come into play
2. **Climate change, fishing, and pollution affect the reef**
   Lorem ipsum dolor sit amet, consectetur adipiscing elit
3. **This area occupies 132,973 mi^2**
   Lorem ipsum dolor sit amet, consectetur adipiscing elit

```python
def getrecs(movie):
#def getrecs(movieId):
    moviestorec = 10

    # Find the movieId of the given movie title
    mask = movies['title_reformatted'].str.upper() == movie.upper()
    movie_df = movies.loc[mask, 'title_reformatted']

    if movie_df.empty:
        print("Please check the spelling of the movie title or the movie may not be in our database :(")
        return

    movie_id = movies.loc[mask, 'movieId'].values
    movie_name = str(movie_df.values[0])  # Convert to string

    if len(movie_id) > 0:
        movie_id = movie_id[0]

        # Check if the movieId exists in the moviesdb DataFrame
        if str(movie_id) in moviesdb['movieId'].values:
            movieindex = moviesdb[moviesdb['movieId'] == str(movie_id)].index
            distances, indices = knn.kneighbors(csr_data[movieindex], n_neighbors=moviestorec + 1)
            recmovie = sorted(list(zip(indices.squeeze().tolist(), distances.squeeze().tolist())), key=lambda x: x[1])[:0:-1
            recframe = []

            for val in recmovie:
                movieindex = moviesdb['movieId'].iloc[val[0]]
                released_year = int(movies[movies['movieId'] == int(movieindex)]['released_year'].values[0])
                recframe.append({
                    'Title': movies[movies['movieId'] == int(movieindex)]['title_reformatted'].values[0],
                    'Released Year': released_year,
                    'Distance': val[1]
                })
            rec_df = pd.DataFrame(recframe, index=range(1, moviestorec + 1))
            rec_df.sort_values(by='Distance', inplace=True)
            rec_df.reset_index(drop=True, inplace=True)
            print(f"If you enjoyed {movie_name} ({released_year}), here are the top 10 movies we think you'll also enjoy!")
            return rec_df
            #return rec_df[["Title", "Released Year"]]
        else:
            return print('There are not enough ratings for this movie.')
    else:
        return print('You get nothing you lose. Good day Sir!')
```

```python
# Example usage of getrecs
getrecs('Rise of the Planet of the Ape')
```

✓ 0.2s

```
Please check the spelling of the movie title or the movie may not be in our database :(
```

```python
getrecs('Rise of the Planet of the Apes')
```

✓ 0.2s

```
If you enjoyed Rise of the Planet of the Apes (2014), here are the top 10 movies we think you'll also enjoy!
```

	Title	Released Year	Distance
0	Dawn of the Planet of the Apes	2014	0.305179
1	Ted	2012	0.392850
2	Mad Max: Fury Road	2015	0.409615
3	Thor	2011	0.427129
4	Star Trek Into Darkness	2013	0.427842
5	X-Men: First Class	2011	0.452307
6	Jurassic World	2015	0.455368
7	The Cabin in the Woods	2012	0.468253
8	The Amazing Spider-Man	2012	0.485955
9	The A-Team	2010	0.486012

```
getrecs('Harry Potter and the Chamber of Secrets')
```

```
If you enjoyed Harry Potter and the Chamber of Secrets (2001), here are the top 10 movies we think you'll also enjoy!
```

	Title	Released Year	Distance
0	Harry Potter and the Sorcerer's Stone (a.k.a. ...	2001	0.196221
1	Harry Potter and the Prisoner of Azkaban	2004	0.208909
2	Harry Potter and the Goblet of Fire	2005	0.265915
3	Harry Potter and the Order of the Phoenix	2007	0.346729
4	Pirates of the Caribbean: Dead Man's Chest	2006	0.349314
5	Pirates of the Caribbean: The Curse of the Bla...	2003	0.367197
6	The Lord of the Rings: The Two Towers	2002	0.391141
7	Harry Potter and the Half-Blood Prince	2009	0.394569
8	Ice Age	2002	0.397131
9	Spider-Man	2002	0.398373

```
getrecs('IRON man')
```

```
If you enjoyed Iron Man (2012), here are the top 10 movies we think you'll also enjoy!
```

	Title	Released Year	Distance
0	The Avengers	2012	0.285319
1	The Dark Knight	2008	0.285835
2	WALL-E	2008	0.298138
3	Iron Man 2	2010	0.307492
4	Avatar	2009	0.310893
5	Batman Begins	2005	0.362759
6	Star Trek	2009	0.366029
7	Watchmen	2009	0.368558
8	Guardians of the Galaxy	2014	0.368758
9	Up	2009	0.368857

```python
# Initialize KNN model
knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20, n_jobs=-1)
knn.fit(csr_data)

# Separate the y variable, the labels
y = ratings['rating']

# Separate the X variable, the features
X = ratings.drop(columns=['rating'])

# Perform feature scaling (standardization)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=42)

# Initialize KNN regressor
knn_regressor = KNeighborsRegressor(n_neighbors=10)

# Fit the model to the training data
knn_regressor.fit(X_train, y_train)

# Make predictions on the test data
predictions = knn_regressor.predict(X_test)

# Calculate and print mean squared error
mse = mean_squared_error(predictions, y_test)
print(f"Mean Squared Error (Accuracy Score): {mse}")
```

Resources

- MovieLens dataset: Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19.
- https://www.geeksforgeeks.org/python-ways-to-sort-a-zipped-list-by-values/#
- https://www.geeksforgeeks.org/numpy-squeeze-in-python/
- https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/#:~:text=The%20K%2DNearest%20Neighbor%20
- https://www.geeksforgeeks.org/user-based-collaborative-filtering/
- https://medium.com/grabngoinfo/recommendation-system-user-based-collaborative-filtering-a2e76e3e15c4
- https://realpython.com/build-recommendation-engine-collaborative-filtering/
- https://www.diva-portal.org/smash/get/diva2:1111865/FULLTEXT01.pdf
- https://movielens.org/
- https://doi.org/10.1145/2827872
- https://www.statology.org/valueerror-unknown-label-type-continuous/
- https://datascience.stackexchange.com/questions/20199/train-test-split-error-found-input-variables-with-inconsistent-numbers-of-sam
- ChatGPT for debugging predictive model assistance

Tableau

https://public.tableau.com/app/profile/shaji.qurashi/viz/Movies_17019978809680/Story1?publish=yes

Website Interface

- Tableau dashboard
- Recommender Demo

WHAT GO HERE

...is where I’d like to go next!