دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده ریاضی و علوم کامپیوتر

پایان‌نامه کارشناسی‌ارشد
گرایش ریاضی

هوش‌مصنوعی - گزارش ۰۶ - پیاده سازی یک سیستم توصیه گر

پایان‌نامه

نگارش
آترین حجت

استاد راهنما
نام کامل استاد راهنما

استاد مشاور
نام کامل استاد مشاور

فروردین ۱۴۰۰


بسم الله الرحمن الرحيم

صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تایید و تصویب پایان نامه موسوم به فرم کمیته دفاع- موجود در پرونده آموزشی- را قرار دهید.

نکات مهم:

- نگارش پایان نامه/رساله باید به زبان فارسی و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان نامه های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت پشت و رو(دورو) بلامانع است و انجام آن توصیه می شود.

دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

به نام خدا

تعهدنامه اصالت اثر

تاریخ: فروردین ۱۴۰۰

اینجانب **آترین حجت** متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

آترین حجت

امضا

فهرست مطالب

عنوان صفحه

فهرست اشکال

شکل صفحه

فهرست جداول

جدول صفحه

فهرست نمادها

نماد	مفهوم
$n(G)$	تعداد رئوس گراف G
$deg_G(v)$	درجه‌ی راس v در گراف G
$nei(v)$	همسایه‌های راس v در گراف

فصل اول

شرح مسئله و روند کار

۱-۱ مقدمه

در این گزارش با استفاده از `filtering Collaborative` یک سیستم توصیه‌گر فیلم می‌سازیم. پیاده سازی و کد در اینجا قابل مشاهده می‌باشد. برای پیاده سازی از Python و کتاب‌خانه‌های numpy, sklearn, scipy, pandas استفاده شده.

۱-۲ روش اجرا

برای اجرای برنامه نیاز به `Python` با حداقل ورژن 3.8 دارید.

برای اجرای برنامه ابتدا نیاز به راه اندازی یک Virtual environment برای `Python` است. به این منظور در فولدر Report 06/Codes دستورات زیر را اجرا کنید.

```
cd Report\\ 06/Codes
python3 -m venv venv
```

برای `activate` کردن با توجه به سیتم عامل دستورات اینجا [۱] را اجرا کنید.

برای نصب پیشنیاز ها دستورات زیر را اجرا کنید.

```
python3 -m pip install -r requirements.txt
```

برای اجرای کد نیاز به دانلود دیتاست دارید [۱] . به صورت پیش‌فرض برنامه درست ../Datasets/ml-latest-small/ به دنبال دیتیست می‌گردد. برای تغییر یک Environment variable با نام `DATASET_FOLDER` درست کنید و مقدار آن را به موقعییت فولدر دیتاست تغییر دهید.

سپس با اجرای فایل GenerateTestValues.py فایل‌های تست و ترینینگ ساخته می‌شوند و می‌توانید فایل ItemToItemCF.py را اجرا کنید.

```
python3 GenerateTestValues.py
python3 ItemToItemCF.py
```

[۱] برای جزئیات بیشتر به اینجا مراجعه کنید

فصل دوم

روش تخمین امتیاز فیلم‌ها

برای تخمین از تکنیک Collaborative Filtering با الگوریتم Nearest Neighbor برای پیداکردن فیلم‌های مشابه استفاده می‌شود.

مرجع فاصله در این گزارش زاویه بین دو بردار در نظر گرفته شده و با تا تابع `cosine` اندازه‌گیری می‌شود. داریم

$$\cos(x, y) = \frac{x.y}{\|x\| \,.\, \|y\|} \tag{2-1}$$

فرض کنید ماتریس `T` حاوی اطلاعات رای دهی کاربران باشد و ماتریس `D` حاوی ضرب داخلی بردارهای امتیازهای هر جفت از فیلم ها باشد. D_{diag} ماتریس مربعی شامل قطر D است.

$$D = T.T^T \tag{2-2}$$

$$\begin{aligned} 1/D_{diag} &= \begin{pmatrix} 1/D_{1,1} & 0 & \cdots & 0 \\ 0 & 1/D_{2,2} & \cdots & - \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/D_{n,n} \end{pmatrix} \\ &= \begin{pmatrix} 1/\|T_1\| & 0 & \cdots & 0 \\ 0 & 1/\|T_2\| & \cdots & - \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\|T_n\| \end{pmatrix} \end{aligned} \tag{2-3}$$

$$\begin{aligned} X &= (1/D_{diag}).(1/D_{diag})^T \\ &= \begin{pmatrix} \frac{1}{\|T_1\|^2} & \frac{1}{\|T_1\|\|T_2\|} & \cdots & \frac{1}{\|T_1\|\|T_n\|} \\ \frac{1}{\|T_2\|\|T_1\|} & \frac{1}{\|T_2\|^2} & \cdots & \frac{1}{\|T_2\|\|T_n\|} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\|T_n\|\|T_1\|} & \frac{1}{\|T_n\|\|T_2\|} & & \cdots \frac{1}{\|T_n\|^2} \end{pmatrix} \end{aligned} \tag{2-4}$$

$$N = \sqrt{X} = \sqrt{(1/D_{diag}).(1/D_{diag})^T}$$
$$= \begin{pmatrix} \frac{1}{\|T_1\|} & \frac{1}{\sqrt{\|T_1\|\|T_2\|}} & \cdots & \frac{1}{\sqrt{\|T_1\|\|T_n\|}} \\ \frac{1}{\sqrt{\|T_2\|\|T_1\|}} & \frac{1}{\|T_2\|} & \cdots & \frac{1}{\sqrt{\|T_2\|\|T_n\|}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{\|T_n\|\|T_1\|}} & \frac{1}{\left\|T_{\surd}\|T_2\|\right\|} & & \cdots \frac{1}{\|T_n\|} \end{pmatrix} \quad (2\text{-}5)$$

حال با ضرب درایه در درایه ماتریس‌های N و D می‌توان فاصله‌های دوبه‌دوی هر دو فیلم را به‌دست آورد.

این بخش در پایتون به صورت زیر به‌دست می‌آید:

```
training_np =training_matrix.to_numpy()
distances =np.matmul(training_np, training_np.transpose())

distances_diagonal =1. /np.diagonal(distances).copy()

norm_size_mul =np.sqrt(np.outer(distances_diagonal, distances_diagonal))

cosine_distances =np.multiply(distances, norm_size_mul)

distances_pd =pd.DataFrame(cosine_distances)
```

۲-۱ پیداکردن Nearest Neighbor

```
NN =NearestNeighbors(n_neighbors=n_neighbors, metric="cosine")

NN.fit(training_csr)

...
```

```
dists, neighbors =NN.kneighbors(movie.values.reshape(1, -1),
                                        n_neighbors=n_neighbors)
```

سپس همسایه‌هایی که user به آنها رای نداده را حذف می‌کنیم.

```
neighbors =[neighbor for neighbor in neighbors[0] if
      training_matrix.iloc[neighbor, userInd] >0]
```

سپس باید با فرمول زیر تخمینی برای امتیاز فیلم به‌دست آوریم.

$$r_{ix} = b_{ix} + \frac{\sum_{j \in N(i;x)} D_{ij}(r_{jx} - bjx)}{\sum_{j \in N(i;x)} D_{ij}} \tag{2-6}$$

که در آن r_{ij} امتیاز فرد j به فیلم i و $b_{ij} = \mu + b_i + b_j$ می‌باشد. μ میانگین همه‌ی امتیازها، b_i فاصله‌ی میانگین امتیازهای فیلم i از میانگین کل، و b_j فاصله‌ی میانگین امتیازهای فرد j از میانگین کل است.

```
user_avg =training.groupby(by="userId").mean()
movie_avg =training.groupby(by="movieId").mean()
global_avg =training.mean()['rating']

bases =(np.ones(training_matrix.values.shape) *-global_avg +
      user_avg.rating.values +
      movie_avg.rating.values.reshape((movie_avg.rating.values.shape[0],
                                       1)))

...

baseline =(global_avg +
      movie_avg.iloc[movie_avg.index==int(row['movieId'])].values[0] +
      user_avg.iloc[user_avg.index==int(row['userId'])].values[0])

neigh_dists_all =distances_pd.iloc[neighbors]
```

```
neigh_dists =neigh_dists_all.iloc[:, movieInd]


bases_nec =bases_pd.iloc[neighbors].loc[:,
       userInd]

ratings_nec =training_matrix.iloc[neighbors].iloc[:,
       training_matrix.columns==int(row['userId'])]



...


ans =bases[movieInd, userInd] +np.inner(ratings_nec.values.reshape(1, -1) -
       bases_nec.values.reshape(1, -1),
       neigh_dists.values.reshape(1, -1)) /np.sum(neigh_dists.values)
```

برای تخمین error حدود 10% از دیتا را ابتدا جدا و به‌عنوان تست استفاده می‌کنیم.

```
import pandas as pd
import numpy as np
import os
# from dotenv import load_dotenv

# load_dotenv()

dataset_folder =os.getenv("DATASET_FOLDER", '../Datasets/ml-latest-small/')


def generate_tests_for_movies():
   ratings =pd.read_csv(os.path.join(dataset_folder, "ratings.csv"))

   users =ratings.groupby(by="userId").count()
```

```
    movies =ratings.groupby(by="movieId").count()

    test_users =users.sample(frac=0.1)
    test_movies =movies.sample(frac=0.1)

    test_ratings =ratings[ratings.userId.isin(test_users.index) &
          ratings.movieId.isin(test_movies.index)]
    training_ratings =ratings[(~ratings.userId.isin(test_users.index)) |
          ~ratings.movieId.isin(test_movies.index)]


    test_ratings.to_csv(os.path.join(dataset_folder, 'ratings.test.csv'))
    training_ratings.to_csv(os.path.join(dataset_folder,
                                         'ratings.training.csv'))

if __name__ =='__main__':
    generate_tests_for_movies()
```

سپس با استفاده از این دیتا خطای مجموع مربعات را محاسبه می‌کنیم.

$$err = \sqrt{\sum_{t \in tests} (r_t - expected_t)^2} \tag{2-7}$$

```
    ...
    err +=(ans -row['rating']) **2
    ...
ans =math.sqrt(err /cnt)
```

کد نهایی به صورت زیر می‌باشد:

```
import pandas as pd
import numpy as np
from sklearn.neighbors import NearestNeighbors
```

```
from scipy.sparse import csr_matrix
import math
import os
# from dotenv import load_dotenv

# load_dotenv()

dataset_folder =os.getenv("DATASET_FOLDER", '../Datasets/ml-latest-small/')

training =pd.read_csv(os.path.join(dataset_folder, "ratings.training.csv"))
test =pd.read_csv(os.path.join(dataset_folder, "ratings.test.csv"))

training_matrix =training.pivot(index="movieId", columns="userId",
      values="rating").fillna(0)
test_matrix =test.pivot(index="movieId", columns="userId",
      values="rating").fillna(0)

training_csr =csr_matrix(training_matrix)

n_neighbors =10
NN =NearestNeighbors(n_neighbors=n_neighbors, metric="cosine")

NN.fit(training_csr)

user_avg =training.groupby(by="userId").mean()
movie_avg =training.groupby(by="movieId").mean()
global_avg =training.mean()['rating']

training_np =training_matrix.to_numpy()
distances =np.matmul(training_np, training_np.transpose())

distances_diagonal =1. /np.diagonal(distances).copy()
```

```
norm_size_mul =np.sqrt(np.outer(distances_diagonal, distances_diagonal))

cosine_distances =np.multiply(distances, norm_size_mul)

distances_pd =pd.DataFrame(cosine_distances)

bases =(np.ones(training_matrix.values.shape) *-global_avg +
       user_avg.rating.values +
       movie_avg.rating.values.reshape((movie_avg.rating.values.shape[0],
                                        1)))

bases_pd =pd.DataFrame(bases)

err, cnt =0, 0
derr, dcnt =0, 0

for index, row in test.iterrows():
   cnt +=1
   if not (training_matrix.index.values==int(row['movieId'])).any():
      print(global_avg, row['rating'])
      err +=(global_avg -row['rating']) **2
      continue
   if not (training_matrix.index.values==int(row['userId'])).any():
      print(global_avg, row['rating'])
      err +=(global_avg -row['rating']) **2
      continue
   movieInd =
                                        np.where(training_matrix.index.values==int(row['mo
   userInd =
                                        np.where(training_matrix.columns.values==int(row['
   movie =training_matrix.iloc[training_matrix.index==int(row['movieId'])]
```

```
dists, neighbors =NN.kneighbors(movie.values.reshape(1, -1),
                                        n_neighbors=n_neighbors)


# for neighbor in neighbors[0]:
#     print(neighbor)

#     print(userInd)
#     print(training_matrix.iloc[neighbor, userInd])
#     print(training_matrix.iloc[neighbor,
#           training_matrix.columns==int(row['userId'])])

neighbors =[neighbor for neighbor in neighbors[0] if
       training_matrix.iloc[neighbor, userInd] >0]

baseline =(global_avg +
       movie_avg.iloc[movie_avg.index==int(row['movieId'])].values[0] +
       user_avg.iloc[user_avg.index==int(row['userId'])].values[0])


if len(neighbors) ==0:
    print(global_avg, row['rating'])
    err +=(global_avg -row['rating']) **2
    continue


neigh_dists_all =distances_pd.iloc[neighbors]
neigh_dists =neigh_dists_all.iloc[:, movieInd]

# print("**********")
# print()
```

```
# print()

# print(row)

# print(neighbors)
# print(neigh_dists)

bases_nec =bases_pd.iloc[neighbors].loc[:,
       userInd]

ratings_nec =training_matrix.iloc[neighbors].iloc[:,
       training_matrix.columns==int(row['userId'])]



ans =bases[movieInd, userInd] +np.inner(ratings_nec.values.reshape(1, -
                                        1) -
       bases_nec.values.reshape(1, -1),
       neigh_dists.values.reshape(1, -1)) /np.sum(neigh_dists.values)

# print("Base", bases[movieInd, userInd])
# print("Rating vec", ratings_nec.values.reshape(1, -1))
# print("Base vec",bases_nec.values.reshape(1, -1))
# print("Distances", neigh_dists.values.reshape(1, -1))
# print("Inner", np.inner(ratings_nec.values.reshape(1, -1) -
#        bases_nec.values.reshape(1, -1),
#        neigh_dists.values.reshape(1, -1)))
# print("Dividend", np.sum(neigh_dists.values))



print(ans, row['rating'])
err +=(ans -row['rating']) **2
derr +=(ans -row['rating']) **2
```

```
    dcnt +=1
    # print()
    # print()

print("Error with mean cases: ", math.sqrt(err /cnt))
print("Error on calculated cases: ", math.sqrt(derr /dcnt))
```

فصل سوم

بررسی دقت با تغییر تعداد همسایه‌ها

به‌منظور بررسی تاثیر تعداد همسایه‌ها از کد TestNumNeighbors.py استفاده شده که میزان خطا را با تغییر تعداد همسایه‌ها می‌سنجد. برای اجرای این کد با درست کردن Environment Variable با نام OUTPUT می‌توان خروجی pgf. jpg. , از نمودار حاصل دریافت کرد.

```
import pandas as pd
import numpy as np
from sklearn.neighbors import NearestNeighbors
from scipy.sparse import csr_matrix
import math
import os
# from dotenv import load_dotenv
from matplotlib.legend_handler import HandlerLine2D, HandlerTuple
import matplotlib
# matplotlib.use("pgf")
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import seaborn as sns


# load_dotenv()

dataset_folder =os.getenv("DATASET_FOLDER", '../Datasets/ml-latest-small/')

training =pd.read_csv(os.path.join(dataset_folder, "ratings.training.csv"))
test =pd.read_csv(os.path.join(dataset_folder, "ratings.test.csv"))

training_matrix =training.pivot(index="movieId", columns="userId",
      values="rating").fillna(0)
test_matrix =test.pivot(index="movieId", columns="userId",
      values="rating").fillna(0)
```

```
training_csr =csr_matrix(training_matrix)



user_avg =training.groupby(by="userId").mean()
movie_avg =training.groupby(by="movieId").mean()
global_avg =training.mean()['rating']

training_np =training_matrix.to_numpy()
distances =np.matmul(training_np, training_np.transpose())

distances_diagonal =1. /np.diagonal(distances).copy()

norm_size_mul =np.sqrt(np.outer(distances_diagonal, distances_diagonal))

cosine_distances =np.multiply(distances, norm_size_mul)

distances_pd =pd.DataFrame(cosine_distances)

bases =(np.ones(training_matrix.values.shape) *-global_avg +
       user_avg.rating.values +
       movie_avg.rating.values.reshape((movie_avg.rating.values.shape[0],
                                        1)))

bases_pd =pd.DataFrame(bases)



dataX, dataErr, dataDErr =[], [], []

for n_neighbors in range(1, 100, 3):
   err, cnt =0, 0
   derr, dcnt =0, 0
```

```
NN =NearestNeighbors(n_neighbors=n_neighbors, metric="cosine")

NN.fit(training_csr)
for index, row in test.iterrows():
    cnt +=1
    if not (training_matrix.index.values==int(row['movieId'])).any():
        # print(global_avg, row['rating'])
        err +=(global_avg -row['rating']) **2
        continue
    if not (training_matrix.index.values==int(row['userId'])).any():
        # print(global_avg, row['rating'])
        err +=(global_avg -row['rating']) **2
        continue
    movieInd =
                                            np.where(training_matrix.index.values==int(row[
    userInd =
                                            np.where(training_matrix.columns.values==int(ro
    movie =
                                            training_matrix.iloc[training_matrix.index==int

    dists, neighbors =NN.kneighbors(movie.values.reshape(1, -1),
                                            n_neighbors=n_neighbors)


    # for neighbor in neighbors[0]:
    #     print(neighbor)

    #     print(userInd)
    #     print(training_matrix.iloc[neighbor, userInd])
    #     print(training_matrix.iloc[neighbor,
    #           training_matrix.columns==int(row['userId'])])
```

```
neighbors =[neighbor for neighbor in neighbors[0] if
      training_matrix.iloc[neighbor, userInd] >0]

baseline =(global_avg +
      movie_avg.iloc[movie_avg.index==int(row['movieId'])].values[0] +
      user_avg.iloc[user_avg.index==int(row['userId'])].values[0])


if len(neighbors) ==0:
   # print(global_avg, row['rating'])
   err +=(global_avg -row['rating']) **2
   continue


neigh_dists_all =distances_pd.iloc[neighbors]
neigh_dists =neigh_dists_all.iloc[:, movieInd]

# print("**********")
# print()
# print()

# print(row)

# print(neighbors)
# print(neigh_dists)

bases_nec =bases_pd.iloc[neighbors].loc[:,
      userInd]

ratings_nec =training_matrix.iloc[neighbors].iloc[:,
      training_matrix.columns==int(row['userId'])]
```

```
    ans =bases[movieInd, userInd] +np.inner(ratings_nec.values.reshape(1,
                                            -1) -
          bases_nec.values.reshape(1, -1),
          neigh_dists.values.reshape(1, -1)) /np.sum(neigh_dists.values)

    # print("Base", bases[movieInd, userInd])
    # print("Rating vec", ratings_nec.values.reshape(1, -1))
    # print("Base vec",bases_nec.values.reshape(1, -1))
    # print("Distances", neigh_dists.values.reshape(1, -1))
    # print("Inner", np.inner(ratings_nec.values.reshape(1, -1) -
    #        bases_nec.values.reshape(1, -1),
    #        neigh_dists.values.reshape(1, -1)))
    # print("Dividend", np.sum(neigh_dists.values))



    # print(ans, row['rating'])
    err +=(ans -row['rating']) **2
    derr +=(ans -row['rating']) **2
    dcnt +=1
    # print()
    # print()

print("**************")
print(f"Neighbors: {n_neighbors}")
print("Error with mean cases: ", math.sqrt(err /cnt))
print("Error on calculated cases: ", math.sqrt(derr /dcnt) if dcnt >0
                                        else 0)
dataX.append(n_neighbors)
dataErr.append(math.sqrt(err /cnt))
dataDErr.append(False)
dataX.append(n_neighbors)
```

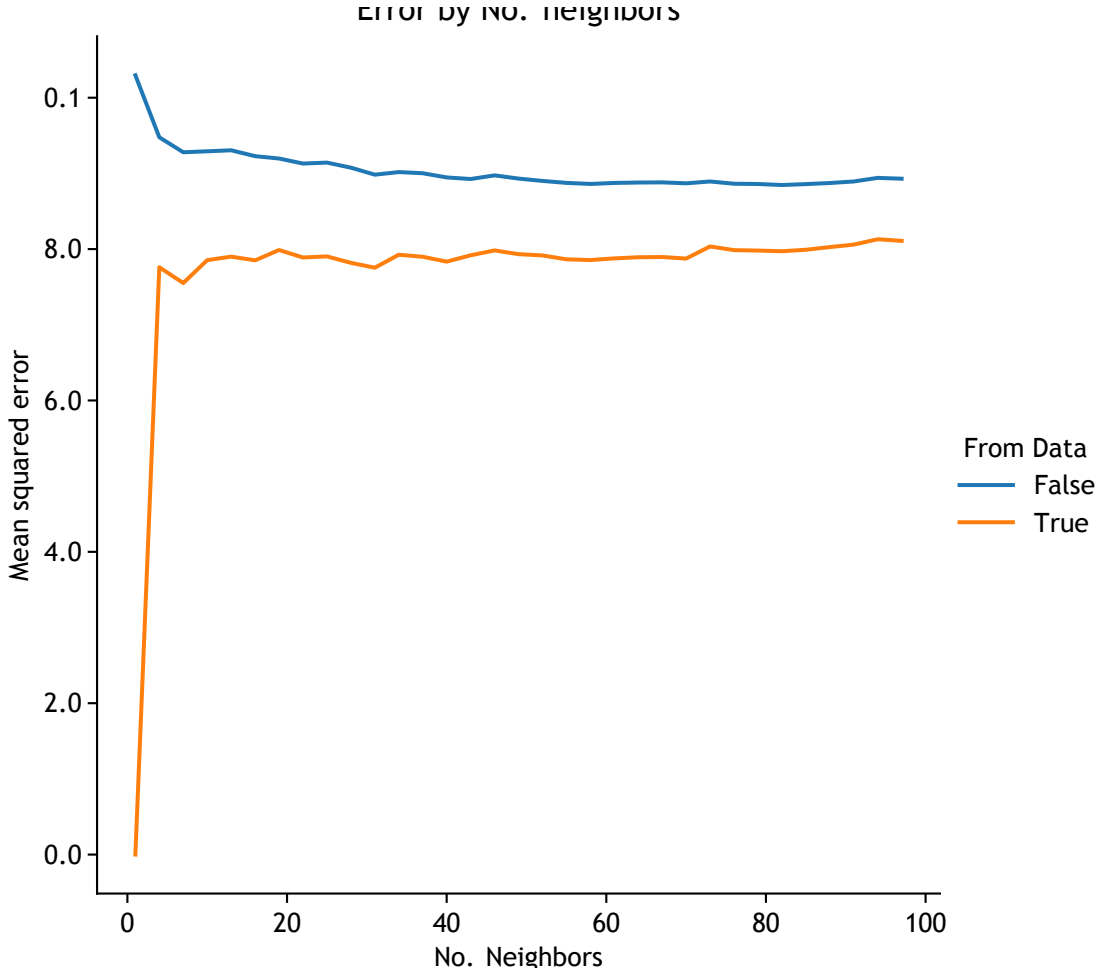
```
    dataErr.append(math.sqrt(derr /dcnt) if dcnt >0 else 0)
    dataDErr.append(True)
    print("**************")

testname ="Error by No. neighbors"
y_lab ="Mean squared error"
x_lab ="No. Neighbors"

plt.figure()
sns.relplot(data=pd.DataFrame({"n_n": dataX, "er": dataErr, "From Data":
                                        dataDErr}), hue="From Data", x="n_n",
                                        y="er", kind="line", )

plt.title(testname)
plt.xlabel(x_lab)
plt.ylabel(y_lab)

if os.getenv("OUTPUT"):
    output =os.getenv("OUTPUT")
    if not os.path.exists(os.path.dirname(os.path.join("./output",
        output, f"{testname}.jpg"))):
        try: os.makedirs(os.path.dirname(os.path.join("./output",
            output, f"{testname}.jpg")))
        except OSError as exc: # Guard against race condition
            if exc.errno !=errno.EEXIST:
                raise
    plt.savefig(os.path.join("./output", output, f"{testname}.jpg"))
    matplotlib.rcParams.update({
        "pgf.texsystem": "xelatex",
        'text.usetex': True,
        'pgf.rcfonts': False,
        "font.family": "mononoki Nerd Font Mono",
```



شکل ۳-۱: Error by No. Neighbors

```
        "font.serif": [],
        # "font.cursive": ["mononoki Nerd Font", "mononoki Nerd Font Mono"],
    })
    plt.savefig(os.path.join("./output", output, f"{testname}.pgf"))

plt.show()
```

در نهایت با بررسی نمودار نهایی تفاوت چندانی بین بررسی ۱۰ تا ۱۰۰ همسایه مشاهده نشد، که البته میتواند به دلیل محدود بودن دیتا باشد. در این نمودار `Data From` فقط دیتاهایی را در نظر می‌گیرد که در همسایه‌هایش این فرد حداقل یک امتیاز داشته باشد. در این شرایط `Data From = False` از میانگین کلی امتیازها استفاده می‌کند.

منابع و مراجع

[1] Harper, F. Maxwell and Konstan, Joseph A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

پیوست