МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А.И. ГЕРЦЕНА»

Направление подготовки

09.03.01 – Информатика и вычислительная техника

Профиль «Технологии разработки программного обеспечения»

Индивидуальная экзаменационная работа по дисциплине Технологии компьютерного моделирования

«Разработка рекомендательной системы с использованием машинного обучения»

Работу выполнил студент 2 курса 2-1 группы:

Воложанин Владислав Олегович

САНКТ-ПЕТЕРБУРГ

2023

Задача

Реализовать с помощью машинного обучения рекомендации мероприятий для пользователей на основе их предпочтений и рейтингов. Есть данные о пользовательских предпочтениях и коэффициентах мероприятий. Задача состоит в том, чтобы использовать модель RandomForestRegressor для предсказания рейтинга каждого мероприятия для каждого пользователя на основе их предпочтений и коэффициентов. Затем необходимо выбрать топ-3 рекомендации для каждого пользователя на основе предсказанных рейтингов, чем выше коэффициент выполнимости, тем с меньшей вероятностью ему предложат мероприятие этого направления.

Первый DataSet (варианты мероприятий)

Датасет хранит информацию о мероприятиях: название и соответствие мероприятия каждому из блоков well being.

event_id	event_name	be_eco	be_friendly	be_fit	be_open	be_pro	be_healthy
1	Научная и учебно-методическая конференция	0	0	0	0	1	0
2	Проектная деятельность проекта	0	0,7	0	0	0,3	0
3	Startup ideas	0	0,2	0	0	0,8	0
4	Выездной лагерь	0	0,9	0	0	0	0,1
5	Halloween	0	1	0	0	0	0
6	Посвящение в студенты Герцена	0	0,8	0	0,2	0	0
7	Культурный квест	0	0,7	0	0	0	0
8	Лига Университета по шахматам	0	0,1	0,8	0	0,1	0
9	Клуб английского	0,6	0,4	0	0	0	0
10	Спорт баттл	0	0,3	0,5	0	0,2	0
11	Студенческий бал	0	1	0	0	0	0
12	Новогодний all styles баттл	0	0,5	0,5	0	0	0

Второй DataSet (данные о пользователях)

Датасет хранит информацию о нескольких юзерах и информации о них:

6 столбцов - количество мероприятий по каждому из направлений, которые пользователь посетил (be eco, be healthy, be fit, be open, be friendly, be pro).

Далее 6 столбцов персональное количество целей юзера по каждому направлению.

Далее 6 столбцов, в которых хранятся числа от 0 до 1 - коэффициент выполненных целей по 6 направлениям.

user_id	NAME	eco	friendly	fit	open	pro	healthy	eco_count	friendly_count	fit_count	open_count	pro_count	healthy_count	Total_events_count	eco_goals	friendly_goals	fit_goals	open_goals
1	Катя	0	0,3	0,2	0,1	0,4	0	3	5	2	1	6	0	17	6	5	3	3
2	Лиза	0,2	0,3	0,5	0	0	0	1	2	3	6	1	1	14	3	3	4	5
3	Маша	0,1	0,1	0,2	0,3	0,1	0,2	6	7	2	5	1	3	24	2	4	5	6
4	Галя	0,2	0,3	0,5	0	0	0	5	1	1	2	3	5	17	4	2	6	3
5	Даша	0,1	0,1	0,2	0,3	0,1	0,2	5	3	2	1	3	5	19	3	6	7	6
6	Женя	0,2	0,3	0,5	0	0	0	5	2	2	1	5	3	18	5	3	8	6
7	Вадик	0,1	0,1	0,2	0,3	0,1	0,2	7	6	3	1	5	3	25	6	2	9	6
8	Катерина	0,2	0,3	0,2	0,1	0,1	0,1	7	8	9	4	5	2	35	2	5	5	6
9	Мария	0,1	0,1	0,2	0,3	0,2	0,1	5	3	2	2	2	2	16	5	6	3	2
10	Даня	0,2	0,3	0,5	0	0	0	1	6	7	8	9	0	31	6	1	2	1
11	Аскар	0,1	0,1	0,2	0,3	0,1	0,2	3	5	7	8	9	0	32	7	1	1	2
12	Эмина	0,2	0,3	0,3	0,1	0,1	0	1	0	0	5	9	0	15	4	2	4	3

pro_goals	healthy_goals	Total Goals	Fillnes	be_eco_pref	be_friendly_pref	be_fit_pref	be_open_pref	be_pro_pref	be_healthy_pref
6	6	29	0,3	1,08	1,17	0,92	0,86	1,26	0,91
3	0	18	0,4	0,84	0,91	1,04	1,31	0,84	0,84
4	3	24	0,2	1,13	1,26	1,09	1,26	1,01	1,01
6	2	23	0,1	1,37	1,05	1,22	1,15	1,34	1,34
2	0	24	0,2	1,19	1,21	1,20	1,10	1,04	1,04
2	1	25	0,6	0,88	0,63	0,83	0,70	0,76	0,76
6	0	29	0,7	0,79	0,61	0,73	0,55	0,71	0,71
7	3	28	0,9	0,37	0,51	0,54	0,43	0,49	0,49
6	2	24	0,1	1,42	1,34	1,15	1,11	1,28	1,28
2	5	17	0,2	1,19	1,05	1,14	1,12	1,21	1,21
1	1	13	0,3	1,33	0,93	1,00	1,10	1,06	1,06
5	0	18	0,5	0,79	0,61	0,72	1,00	1,38	1,38

Обзор кода

Импортированные библиотеки

1. *pandas* – для манипулирования данными.

2. *train_test_split* – разделения набора данных на обучающий и тестовый наборы.

3. *mean_squared_error* – для оценки производительности модели.

4. *RandomForestRegressor* – для создания рекомендательной модели.

5. *numpy* – для численных вычислений.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
import numpy as np
```

Данные о событиях

events_data содержит информацию о различных событиях.

```
events_data = {
    'event_id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
    'be_eco': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'be_friendly': [0, 0.7, 0.2, 0.9, 1, 0.8, 0.7, 0.1, 0.4, 0.3, 1, 0.5],
    'be_fit': [0, 0, 0, 0, 0, 0, 0, 0.8, 0, 0.5, 0, 0.5],
    'be_open': [0, 0, 0, 0, 0, 0.2, 0, 0, 0, 0, 0, 0],
    'be_pro': [1, 0.3, 0.8, 0, 0, 0, 0, 0.1, 0, 0.2, 0, 0],
    'be_healthy': [0, 0, 0, 0.1, 0, 0, 0, 0, 0, 0, 0, 0]
}
```

Данные о пользователях

users_data содержит информацию о разных пользователях.

```
users_data = {
    'user_id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
    'be_eco_pref': [1.08, 0.84, 1.13, 1.37, 1.19, 0.88, 0.79, 0.37, 1.42, 1.19, 1.33, 0.79],
    'be_friendly_pref': [1.17, 0.91, 1.26, 1.05, 1.21, 0.63, 0.61, 0.51, 1.34, 1.05, 0.93, 0.61],
    'be_fit_pref': [0.92, 1.04, 1.09, 1.22, 1.20, 0.83, 0.73, 0.54, 1.15, 1.14, 1.00, 0.72],
    'be_open_pref': [0.86, 1.31, 1.26, 1.15, 1.10, 0.7, 0.55, 0.43, 1.11, 1.12, 1.1, 1],
    'be_pro_pref': [1.26, 0.84, 1.01, 1.34, 1.04, 0.76, 0.71, 0.49, 1.28, 1.21, 1.06, 1.38],
    'be_healthy_pref': [0.91, 0.84, 1.01, 1.34, 1.04, 0.76, 0.71, 0.49, 1.28, 1.21,  1.06, 1.38]
}
```

Создаем DataSet для обучения

Формирование датасета для обучения модели. Мы создаем пустой список *dataset* и итерируем по пользователям и событиям, чтобы вычислить рейтинг каждого события для каждого пользователя на основе их предпочтений и коэффициентов событий. Полученные данные добавляются в список *dataset*.

```
dataset = []
for _, user in users_df.iterrows():
    user_id = user['user_id']
    user_preferences = user[['be_eco_pref', 'be_friendly_pref', 'be_fit_pref',
                             'be_open_pref', 'be_pro_pref', 'be_healthy_pref']].values

    for _, event in events_df.iterrows():
        event_id = event['event_id']
        event_coefficients = event[['be_eco', 'be_friendly', 'be_fit',
                                    'be_open', 'be_pro', 'be_healthy']].values

        rating = np.dot(user_preferences, event_coefficients)

        dataset.append((user_id, event_id, rating))
```

Создаем DataFrame из датасета

Словари преобразуются в фреймы данных pandas с помощью функции *pd.DataFrame()*. Фреймы данных печатаются для отображения данных.

```
dataset_df = pd.DataFrame(dataset, columns=['user_id', 'event_id', 'rating'])
```

Сохраняем DataSet в CSV-файл

Набор данных преобразованный в DataFrame, сохраняется в CSV-файл с именем "recommendation_dataset.csv" с помощью метода *to_csv*.

```
dataset_df.to_csv('recommendation_dataset.csv', index=False)
```

Загрузка датасета

Файл CSV загружается в новый фрейм данных с помощью *pd.read_csv()*.

```
dataset_df = pd.read_csv('recommendation_dataset.csv')
```

Разделение датасета на обучающую и тестовую выборки.

Набор данных разбивается на обучающий и тестовый наборы с помощью функции *train_test_split()* из *sklearn.model_selection*. Обучающий набор используется для обучения рекомендательной модели, а тестовый набор будет использоваться для оценки производительности модели.

```
train_df, test_df = train_test_split(dataset_df, test_size=0.2, random_state=42)
```

Формирование матрицы признаков и целевой переменной

Формирование матрицы признаков *X_train* и *X_test*, содержащих столбцы 'user_id' и 'event_id', а также целевой переменной *y_train* и *y_test*, содержащих столбец 'rating'.

```
X_train = train_df[['user_id', 'event_id']].values
y_train = train_df['rating'].values

X_test = test_df[['user_id', 'event_id']].values
y_test = test_df['rating'].values
```

Обучение модели рекомендательной системы

Модель регрессии случайного леса создается со 100 оценками с использованием *RandomForestRegressor()* из *sklearn.ensemble*. Затем модель обучается на основе обучающих данных (*X_train*) и (*y_train*), с использованием метода *fit()*.

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Прогнозирование рейтингов для тестовой выборки

Модель используется для прогнозирования рейтингов на тестовой выборке *X_test*, и результаты сохраняются в *y_pred* с использованием метода *predict()*.

```
y_pred = model.predict(X_test)
```

Вычисление среднеквадратичной ошибки

Среднеквадратичная ошибка (MSE) вычисляется путем сравнения прогнозируемых оценок *(y_pred)* с фактическими оценками (y_test) с использованием функции *mean_squared_error()* из *sklearn.metrics*. MSE - это показатель точности модели, выводится на консоль.

```
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)
```

Получения топ-3 рекомендаций для каждого пользователя

Эта часть кода демонстрирует пример получения 3 лучших рекомендаций для каждого пользователя. Он перебирает уникальные идентификаторы пользователей, извлекает их предпочтения из фрейма данных *users_df* и прогнозирует рейтинги для всех событий. В качестве рекомендаций выбираются 3 лучших события с самыми высокими прогнозируемыми рейтингами. Рекомендации выводятся на консоль с указанием идентификаторов событий и соответствующих им оценок для каждого пользователя.

```
unique_users = dataset_df['user_id'].unique()

for user_id in unique_users:
    user_preferences = users_df[users_df['user_id'] == user_id][['be_eco_pref', 'be_friendly_pref', 'be_fit_pref',
                                                                 'be_open_pref', 'be_pro_pref', 'be_healthy_pref']].values
    event_ids = events_df['event_id'].values
    predictions = []

    for event_id in event_ids:
        X_pred = np.array([[user_id, event_id]])
        rating_pred = model.predict(X_pred)
        predictions.append((event_id, rating_pred))

    top_recommendations = sorted(predictions, key=lambda x: x[1], reverse=True)[:3]

    print("Рекомендации для пользователя", user_id)
    for recommendation in top_recommendations:
        event_id, rating = recommendation
        print("Event_id", event_id, "- Рейтинг:", rating)
    print()
```

Вывод программы

Первая часть вывода программы: программа выводит на экран 2 датасета (данные о пользователях, варианты мероприятия).

	event_id	be_eco	be_friendly	be_fit	be_open	be_pro	be_healthy
0	1	0	0.0	0.0	0.0	1.0	0.0
1	2	0	0.7	0.0	0.0	0.3	0.0
2	3	0	0.2	0.0	0.0	0.8	0.0
3	4	0	0.9	0.0	0.0	0.0	0.1
4	5	0	1.0	0.0	0.0	0.0	0.0
5	6	0	0.8	0.0	0.2	0.0	0.0
6	7	0	0.7	0.0	0.0	0.0	0.0
7	8	0	0.1	0.8	0.0	0.1	0.0
8	9	0	0.4	0.0	0.0	0.0	0.0
9	10	0	0.3	0.5	0.0	0.2	0.0
10	11	0	1.0	0.0	0.0	0.0	0.0
11	12	0	0.5	0.5	0.0	0.0	0.0

	user_id	be_eco_pref	be_friendly_pref	be_fit_pref	be_open_pref	be_pro_pref	be_healthy_pref
0	1	1.08	1.17	0.92	0.86	1.26	0.91
1	2	0.84	0.91	1.04	1.31	0.84	0.84
2	3	1.13	1.26	1.09	1.26	1.01	1.01
3	4	1.37	1.05	1.22	1.15	1.34	1.34
4	5	1.19	1.21	1.20	1.10	1.04	1.04
5	6	0.88	0.63	0.83	0.70	0.76	0.76
6	7	0.79	0.61	0.73	0.55	0.71	0.71
7	8	0.37	0.51	0.54	0.43	0.49	0.49
8	9	1.42	1.34	1.15	1.11	1.28	1.28
9	10	1.19	1.05	1.14	1.12	1.21	1.21
10	11	1.33	0.93	1.00	1.10	1.06	1.06
11	12	0.79	0.61	0.72	1.00	1.38	1.38

Вторая часть вывода программы: для каждого из пользователей программа выдает 3 варианта мероприятия по его предпочтениям.

```
Mean Squared Error: 0.016950767679310 35
Рекомендации для пользователя 1.0
Event_id 1 - Рейтинг: [1.23864]
Event_id 3 - Рейтинг: [1.21613]
Event_id 2 - Рейтинг: [1.2088]

Рекомендации для пользователя 2.0
Event_id 6 - Рейтинг: [0.97058]
Event_id 12 - Рейтинг: [0.96854]
Event_id 1 - Рейтинг: [0.96315]

Рекомендации для пользователя 3.0
Event_id 5 - Рейтинг: [1.2289]
Event_id 6 - Рейтинг: [1.22788]
Event_id 11 - Рейтинг: [1.2074]

Рекомендации для пользователя 4.0
Event_id 1 - Рейтинг: [1.23281]
Event_id 3 - Рейтинг: [1.22908]
Event_id 2 - Рейтинг: [1.18241]

Рекомендации для пользователя 5.0
Event_id 12 - Рейтинг: [1.19178]
Event_id 11 - Рейтинг: [1.18982]
Event_id 5 - Рейтинг: [1.18656]

Рекомендации для пользователя 6.0
Event_id 8 - Рейтинг: [0.75657]
Event_id 1 - Рейтинг: [0.73497]
Event_id 7 - Рейтинг: [0.71818]
```

```
Рекомендации для пользователя 7.0
Event_id 1 - Рейтинг: [0.69935]
Event_id 3 - Рейтинг: [0.68288]
Event_id 2 - Рейтинг: [0.68101]

Рекомендации для пользователя 8.0
Event_id 12 - Рейтинг: [0.52561]
Event_id 11 - Рейтинг: [0.52349]
Event_id 5 - Рейтинг: [0.50675]

Рекомендации для пользователя 9.0
Event_id 3 - Рейтинг: [1.29016]
Event_id 11 - Рейтинг: [1.28711]
Event_id 1 - Рейтинг: [1.2848]

Рекомендации для пользователя 10.0
Event_id 1 - Рейтинг: [1.17101]
Event_id 2 - Рейтинг: [1.10735]
Event_id 3 - Рейтинг: [1.0973]

Рекомендации для пользователя 11.0
Event_id 1 - Рейтинг: [1.05777]
Event_id 3 - Рейтинг: [1.0281]
Event_id 2 - Рейтинг: [0.99922]

Рекомендации для пользователя 12.0
Event_id 3 - Рейтинг: [1.03174]
Event_id 1 - Рейтинг: [1.02549]
Event_id 2 - Рейтинг: [1.01505]
```

Заключение

В данном задании была реализована модель машинного обучения на основе алгоритма RandomForestRegressor для предсказания рейтингов мероприятий для каждого пользователя на основе их предпочтений и коэффициентов.

Для достижения этой цели был проведен ряд шагов. Вначале были подготовлены исходные данные о пользовательских предпочтениях и коэффициентах мероприятий. Затем эти данные были использованы для обучения модели RandomForestRegressor.

Были получены предсказания рейтингов для всех мероприятий и пользователей. Затем был выполнен этап выбора топ-3 рекомендаций для каждого пользователя на основе предсказанных рейтингов. Этот шаг позволил определить наиболее подходящие мероприятия для каждого пользователя.

В результате работы модели была создана система рекомендаций, которая способна предложить каждому пользователю топ-3 мероприятия.

Код

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
import numpy as np
events_data = {
    'event_id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
    'be_eco': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'be_friendly': [0, 0.7, 0.2, 0.9, 1, 0.8, 0.7, 0.1, 0.4, 0.3, 1, 0.5],
    'be_fit': [0, 0, 0, 0, 0, 0, 0, 0.8, 0, 0.5, 0, 0.5],
    'be_open': [0, 0, 0, 0, 0, 0.2, 0, 0, 0, 0, 0, 0],
    'be_pro': [1, 0.3, 0.8, 0, 0, 0, 0, 0.1, 0, 0.2, 0, 0],
    'be_healthy': [0, 0, 0, 0.1, 0, 0, 0, 0, 0, 0, 0, 0]
}
events_df = pd.DataFrame(events_data)
print(events_df)
users_data = {
    'user_id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
    'be_eco_pref': [1.08, 0.84, 1.13, 1.37, 1.19, 0.88, 0.79, 0.37, 1.42, 1.19, 1.33, 0.79],
    'be_friendly_pref': [1.17, 0.91, 1.26, 1.05, 1.21, 0.63, 0.61, 0.51, 1.34, 1.05, 0.93,
0.61],
    'be_fit_pref': [0.92, 1.04, 1.09, 1.22, 1.20, 0.83, 0.73, 0.54, 1.15, 1.14, 1.00, 0.72],
    'be_open_pref': [0.86, 1.31, 1.26, 1.15, 1.10, 0.7, 0.55, 0.43, 1.11, 1.12, 1.1, 1],
    'be_pro_pref': [1.26, 0.84, 1.01, 1.34, 1.04, 0.76, 0.71, 0.49, 1.28, 1.21, 1.06, 1.38],
    'be_healthy_pref': [0.91, 0.84, 1.01, 1.34, 1.04, 0.76, 0.71, 0.49, 1.28, 1.21,  1.06,
1.38]
}
users_df = pd.DataFrame(users_data)
print(users_df)
dataset = []
for _, user in users_df.iterrows():
    user_id = user['user_id']
    user_preferences = user[['be_eco_pref', 'be_friendly_pref', 'be_fit_pref',
                             'be_open_pref', 'be_pro_pref', 'be_healthy_pref']].values
    for _, event in events_df.iterrows():
        event_id = event['event_id']
```

```
        event_coefficients = event[['be_eco', 'be_friendly', 'be_fit',
                                    'be_open', 'be_pro', 'be_healthy']].values
        rating = np.dot(user_preferences, event_coefficients)
        dataset.append((user_id, event_id, rating))
dataset_df = pd.DataFrame(dataset, columns=['user_id', 'event_id', 'rating'])
dataset_df.to_csv('recommendation_dataset.csv', index=False)
dataset_df = pd.read_csv('recommendation_dataset.csv')
train_df, test_df = train_test_split(dataset_df, test_size=0.2, random_state=42)
X_train = train_df[['user_id', 'event_id']].values
y_train = train_df['rating'].values
X_test = test_df[['user_id', 'event_id']].values
y_test = test_df['rating'].values
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)
unique_users = dataset_df['user_id'].unique()
for user_id in unique_users:
    user_preferences = users_df[users_df['user_id'] == user_id][['be_eco_pref',
'be_friendly_pref', 'be_fit_pref',
                                                                 'be_open_pref',
'be_pro_pref', 'be_healthy_pref']].values
    event_ids = events_df['event_id'].values
    predictions = []
    for event_id in event_ids:
        X_pred = np.array([[user_id, event_id]])
        rating_pred = model.predict(X_pred)
        predictions.append((event_id, rating_pred))
    top_recommendations = sorted(predictions, key=lambda x: x[1], reverse=True)[:3]
    print("Рекомендации для пользователя", user_id)
    for recommendation in top_recommendations:
        event_id, rating = recommendation
        print("Event_id", event_id, "- Рейтинг:", rating)
    print()
```

Список использованных источников

1. Hands-On Machine Learning with Scikit-Learn and TensorFlow / Aurelien Geron.
2. Python Machine Learning / Sebastian Raschka, Vahid Mirjalili.