Большие данные и машинное обучение

Инженерный тур

Легенда задачи

- Есть лента «новостей» в некоторой социальной сети, которая наполнена различным контентом.
- Контент состоит из множества записей, которые показываются пользователю.
- Пользователь просматривает каждую запись и как-то взаимодействует ней.
- Для построения системы рекомендации требуется предсказывать тип взаимодействия.

Описание оборудования и программного обеспечения

Участникам предоставляется доступ к виртуальному серверу для проведения вычислений:

- 12 ГБ ОЗУ;
- 40 Гб жесткий диск;
- 4 ядра процессора;
- ОС Ubuntu 16.04+.

На сервере установлен Python 3.9 с Jupyter Notebook, SciPy, NumPy, Scikit Learn и Pandas. Участники могут доустановить необходимые им библиотеки и языки.

Доступ к серверу осуществляется с личного или выданного ноутбука через веб-интерфейс Jupyter Notebook.

Решения загружаются на платформе Codeforces.

Описание этапов работы участников

1. Знакомство с задачей.
2. Разработка и реализация подхода к работе с данными, которые хранятся в нескольких таблицах.
3. Разработка программы, которая построит модель предсказания.
4. Поиск оптимальных гиперпараметров выбранной модели предсказания.
5. Построение предсказания на тестовом множестве и загрузка его в тестирующую систему.

Задача VI.2.3.1. Предсказание оценки (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 15 с.

Ограничение по памяти: 256 Мбайт.

Условие

Требуется предсказать, понравится ли пользователям запись в социальной сети?

Формат входных данных

Набор данных, подготовленный командой VK, доступен по адресу https://disk.yandex.ru/d/Gffq-VhB5jekKQ.

Он состоит из нескольких CSV файлов:

- `topics.csv` — содержит информацию о записях социальной сети «Одноклассники». Каждая запись характеризуется признаками текста и изображения.
- `users.csv` — содержит информацию о пользователях социальной сети «Одноклассники». Каждый пользователь характеризуется датой рождения, полом и идентификатором города.
- `train.csv` — содержит информацию о взаимодействии пользователей с записями социальной сети «Одноклассники». Тип взаимодействия равен L, если запись понравилась, и D — если запись не понравилась.
- `test.csv` — содержит идентификаторы записей и пользователей, для которых требуется предсказать тип взаимодействия.

Формат выходных данных

Необходимо загрузить в тестирующую систему текстовый файл с ответами. Он не должен содержать заголовки и никакой другой дополнительной информации. Каждая строка должна содержать одну заглавную латинскую букву D или L. Ответы должны быть даны в порядке, в котором были соответствующие запросы из файла `test.csv`.

Вы можете совершить **не более 50-ти попыток** решения данной задачи.

Критерии оценивания

Для оценки будет использоваться F-мера класса L, умноженная на 10000.

Результат, который вы будите получать во время соревнования, будет вычислен только на 20% тестовой выборки. В конце соревнования выбранные вами решения **будут перетестированы** на оставшихся 80%.

Решение

Задачу можно решить несколькими способами.

В качестве базового решения используется подход, в котором для каждой записи вычисляется число отметок типа L (нравится) и D (не нравится). Если отметок L было больше, то для всех запросов из тестового множества будет даваться ответ L независимо от пользователя, иначе будет даваться ответ D.

Можно решить эту задачу как классическую задачу обучения с учителем. Для этого требуется объединить таблицы с описанием пользователей и записей. Но в результате будут получаться объекты из большого числа признаков. Для построения модели можно воспользоваться библиотекой XGBoost.

Также можно решить эту задачу как классическую задачу коллаборативной фильтрации. Для этого достаточно использовать только таблицу с информацией о взаимодействии пользователей и записей. Можно представить, что эта таблица кодирует разреженную матрицу размера N на M, где N — число пользователей, а M — число записей. Взаимодействие L можно кодировать значением $+1$, а D — значением -1. В таком виде можно попытаться факторизовать матрицу на две матрицы с небольшим рангом. Например, для этого можно использовать метод `NMF` из библиотеки `surprise`.

Помимо этого можно объединить предыдущие два решения, если сконкатенировать полученные при разложении матриц признаки с данным признаками пользователей и записей.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import pandas as pd
import numpy as np
import pickle
from sklearn.model_selection import KFold
from sklearn.metrics import f1_score
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
OPTIMIZE_ROUNDS = False

from catboost import cv, Pool, CatBoostClassifier

train = pd.read_csv('/kaggle/input/nonononono/trainavrdd.csv')
test = pd.read_csv('/kaggle/input/nonononono/testavrdd.csv')

train.birthDate.fillna(train.birthDate.median(), inplace = True)
train.drop('Unnamed: 0', axis = 1, inplace = True)
test.drop('Unnamed: 0', axis = 1, inplace = True)

train['locationId']=train['locationId'].astype("category")
train['gender']=train['gender'].astype("category")
train['topicId']=train['topicId'].astype("category")
train['userId']=train['userId'].astype("category")
test['locationId']=test['locationId'].astype("category")
test['gender']=test['gender'].astype("category")
test['topicId']=test['topicId'].astype("category")
test['userId']=test['userId'].astype("category")

train['birthDate'] = (train.birthDate - train.birthDate.min()) /
↪  (train.birthDate.max() - train.birthDate.min())
train['countus'] = (train.countus - train.countus.min()) / (train.countus.max() -
↪  train.countus.min())
train['counttop'] = (train.counttop - train.counttop.min()) /
↪  (train.counttop.max() - train.counttop.min())

test.rename({'countus_y':'countus'}, axis=1)

```

```
test['birthDate'] = (test.birthDate - test.birthDate.min()) / (test.birthDate.max() - test.birthDate.min())
test['countus_y'] = (test.countus_y - test.countus_y.min()) / (test.countus_y.max() - test.countus_y.min())
test['counttop'] = (test.counttop - test.counttop.min()) / (test.counttop.max() - test.counttop.min())

train.drop(['Unnamed: 0.1', 'locationId', 'userId', 'topicId'], axis=1, inplace=True)

y = train['feedback']
X = train.drop('feedback', axis = 1)

y = y.replace([-1], 0)

K = 5
kf = KFold(n_splits = K, random_state = 1, shuffle = True)

model = CatBoostClassifier(
    cat_features = ['gender'],
    depth = 16,
    l2_leaf_reg = 34,
    random_seed=42,
    learning_rate=0.00009,
    iterations = 50,
    loss_function='Logloss'
)

for i, (train_index, test_index) in enumerate(kf.split(train)):
    y_train, y_valid = y.iloc[train_index], y.iloc[test_index]
    X_train, X_valid = X.iloc[train_index,:], X.iloc[test_index,:]
    print( "\nFold ", i)
    if OPTIMIZE_ROUNDS:
        fit_model = model.fit( X_train, y_train,
                               eval_set=[X_valid, y_valid],
                               use_best_model=True
                             )

    else:
        fit_model = model.fit( X_train, y_train )

test.drop(['Unnamed: 0.1', 'locationId', 'userId', 'topicId'], axis=1,inplace=True)
model.get_feature_importance(prettified=True)
test.drop(['Unnamed: 0.1', 'locationId', 'userId', 'topicId'], axis=1,inplace=True)

f=(model.predict(test))

d1=[]
for i in f:
    if i == 1:
        d1.append('L')
    else:
        d1.append('D')
len(d1)

g1=True
for i in d1:
    if g1==True:
```

```
89          my_file = open("ff12.txt", "w+")
90          my_file.write(i)
91          my_file.write('\n')
92          my_file.close()
93          g1=False
94      else:
95          my_file = open("ff12.txt", "a+")
96          my_file.write(i)
97          my_file.write('\n')
98          my_file.close()
```

Материалы для подготовки

- https://github.com/deeppavlov/dlschl.
- https://stepik.org/course/1296/promo.
- https://stepik.org/course/217/promo.
- https://stepik.org/course/76/promo.
- https://drive.google.com/file/d/1i1_Wf8jiWd3o0nRZGaX2gdoHCTIkvKOJ/view.