Второй отборочный этап

Индивидуальная часть

Задача II.1.1. Капитальный ремонт (100 баллов)

Вове нужно сделать ремонт на даче. Есть X мастеров, которые умеют делать разные работы. Есть список работ, который нужно сделать Вове. Каждый раз когда новые мастер приходит продолжать работу, он ругает предыдущего мастера и говорит, и много возмущается. Необходимо составить расписание работ мастеров так, чтобы количество возмущений (перемен с одного мастера на другого) было минимальным. У каждого мастера список того, что он умеет. В качестве ответа предоставляется минимальное число перемен мастеров и последовательность мастеров. Если таких последовательностей несколько, то подойдет любая из них в качестве правильного ответа.

Ответ представляется в виде списка имен мастеров, которые делают работу. Длина списка имен равна длине списка работ.

```
Имена мастеров =
↪  ["Антон","Егор","Саша","Леша","Женя","Витя","Коля","Кеша","Стас","Миша"]

Список работ = ["пилить", "рубить", "таскать","сверлить",  "носить", "вытачивать",
↪  "красить","лакировать","вешать шторы", "укладывать пол", "ставить окна","вешать
↪  люстру",   "строить крыльцо","закупать материалы"]
```

Количество мастеров от 6 до 10 (у всех мастеров разные имена) Количество навыков мастера от 3 до 6 (навыки не повторяются) Количество работ от 5 до 15 (работы могут повторяться).

Если работы текущими мастерами выполнить невозможно, то надо вернуть пустой лист [].

Формат запроса ([(имя_работника,[список_работ],..)], [список_работ]) формат ответа [имя_работника,. . .]

для ответа вам нужно загрузить файл submission.py в котором есть функция eval(data), при этом в переменную data передается запрос. пример:

submission.py:

```
def eval(data):
    return [data[0][0][0],data[0][1][0]]
```

Если ваш код падает, то возможно посмотреть ошибки. Для решения задачи, можно пользоваться только стандартными библиотеками.

будет в качестве ответа возвращать имена двух работников (ответ не правильный, но корректный с точки зрения исполнения кода).

Пример запроса: ([(‘Антон’, [‘строить крыльцо’, ‘ставить окна’, ‘рубить’]), (‘Егор‘, [‘таскать’, ‘вешать люстру’, ‘красить’, ‘строить крыльцо’]), (‘Саша’, [‘лакировать’, ‘вешать шторы’, ‘сверлить’]), (‘Леша’, [‘закупать материалы’, ‘рубить’, ‘пилить’, ‘таскать’, ‘ставить окна’]), (‘Женя’, [‘вешать шторы’, ‘сверлить’, ‘таскать’, ‘лакировать’]), (‘Витя’, [‘закупать материалы’, ‘строить крыльцо’, ‘сверлить’, ‘пилить’, ‘укладывать пол’, ‘лакировать’]), (‘Коля’, [‘носить’, ‘вытачивать’, ‘красить’, ‘таскать’, ‘ставить окна’])], [‘закупать материалы’, ‘пилить’, ‘пилить’, ‘сверлить’, ‘рубить’, ‘пилить’, ‘красить’, ‘носить’, ‘пилить’]) Ответ: [‘Витя’, ‘Витя’, ‘Витя’, ‘Витя’, ‘Леша’, ‘Леша’, ‘Коля’, ‘Коля’, ‘Леша’]

В качестве теста решение будет проверено на 30 запусках, если все запуски решены верно, то получаете 100 баллов, если есть 1 ошибка, то 50 баллов, если более одной ошибки, то 0 баллов. В рамках решения данной задачи количество пакетов ограниченно стандартными пакетами python3, pandas, scikit-learn, scipy.

Решение

```
import ast
def eval(data):
    return [data[0][0][0],data[0][1][0]]

# def eval(data):
# #    d = ast.literal_eval(data)
#     result=resolve(*data)
#     return str(result[1])



def resolve(workers_list, jobs, worker=""):
    results = []
    if len(jobs) == 0:
        return 0,[]
    if worker !="":
        cur_worker = [w[1] for w in workers_list if w[0]==worker][0]
        if jobs[0] in cur_worker:
            res = resolve(workers_list, jobs[1:], worker=worker)
            return res[0],[worker]+res[1]
    for w in workers_list:
        if jobs[0] in w[1]:
            res = resolve(workers_list, jobs[1:], worker=w[0])
            if res[0]==-1:
                continue
            penalty = res[0]
            if w[0]!=worker and worker!="":
                penalty+=1
            results.append((penalty,[w[0]]+res[1]))
    if len(results)==0:
        return -1,[]
    return min(results,key=lambda x: x[0])


def resolve_r(workers_list, jobs, worker=""):
    results = []
    if len(jobs) == 0:
        return 0,[]
    if worker !="":
        cur_worker = [w[1] for w in workers_list if w[0]==worker][0]
```

```
        if jobs[0] in cur_worker:
            res = resolve(workers_list, jobs[1:], worker=worker)
            return res[0],[worker]+res[1]
    for w in workers_list:
        if jobs[0] in w[1]:
            res = resolve(workers_list, jobs[1:], worker=w[0])
            if res[0]==-1:
                continue
            penalty = res[0]
            if w[0]!=worker and worker!="":
                penalty+=1
            results.append((penalty,[w[0]]+res[1]))
    if len(results)==0:
        return -1,[]
    return results[0]
```

Задача II.1.2. Помощь парашютисту (500 баллов)

Парашютист выпрыгнул из самолета, но у него парашют раскрылся неполностью. К счастью рядом оказался спасательный дрон. Для спасения дрону необходимо рассчитать траекторию падающего парашютиста. Координата высоты вычисляема тривиально, однако дрону необходимо вычислить плоскостные координаты (x, y). Есть траектория изменения, по ним необходимо вычислить последующие координаты дрон.

Есть примерная формула изменения координат, она зависит от двух коэффициентов $k1$ и $k2$ и случайного шума координата $x = k2cos(k1t) +$ шум координата $y = k1sin(k2t) +$ шум.

На вход подается list в 1000 пар координат x y первые тысячу секунд (с 0 по 999ю):

```
[(0.8990863593574939, 8.475387292276096e-07), (0.8089593833113505,
↪  0.3534820061950688), (0.5566465706027002, 0.4399620826711653),
↪  (0.19273411906364982, 0.19411748737324), (-0.20981822525965102,
↪  -0.19835273653070376), (-0.5703058016775612, -0.44099688186601116),
↪  (-0.8164537877967621, -0.35053466833726243), (-0.8989145629777989,
↪  0.004703700279075586), ... (-0.801155143314037, 0.35638955826565816),
↪  (-0.5427749453575453, 0.43887713883486584), (-0.17557562076019273,
↪  0.18985994842390025), (0.22682425709951057, -0.20256735021079741),
↪  (0.5837487993279974, -0.4419853288796921), (0.8236405373374047,
↪  -0.3475501356746985), (0.8984026035824828, 0.009406628459256976),
↪  (0.7930484451922395, 0.3592589441057791)]
```

В ответ ожидается массив в 1000 значений пар на вторую тысячу секунд (с 1000й по 1999ю), пример:

```
[(0.8990863593574939, 8.475387292276096e-07), (0.8089593833113505,
↪  0.3534820061950688), (0.5566465706027002, 0.4399620826711653),
↪  (0.19273411906364982, 0.19411748737324), (-0.20981822525965102,
↪  -0.19835273653070376), (-0.5703058016775612, -0.44099688186601116),
↪  (-0.8164537877967621, -0.35053466833726243), (-0.8989145629777989,
↪  0.004703700279075586), ... (-0.801155143314037, 0.35638955826565816),
↪  (-0.5427749453575453, 0.43887713883486584), (-0.17557562076019273,
↪  0.18985994842390025), (0.22682425709951057, -0.20256735021079741),
↪  (0.5837487993279974, -0.4419853288796921), (0.8236405373374047,
↪  -0.3475501356746985), (0.8984026035824828, 0.009406628459256976),
↪  (0.7930484451922395, 0.3592589441057791)]
```

Ответ необходимо загрузить в файл submission.py Пример функции ответа:

```
import random
def eval(data):
    return [(random.random(),random.random()) for i in range(1000,2000)]
```

Решение

```
import random
import numpy as np
import sklearn
import math
# from sklearn.tree import DecisionTreeRegressor
# from lightgbm import LGBMRegressor
from sklearn.linear_model import LinearRegression
def gen_data1(x):
    return list(zip(x,[math.cos(a) for a in x],[math.sin(a) for a in x],[math.cos(2*a)
    ↪  for a in x],[math.sin(2*a) for a in x]))

def eval2(data):
    return [(random.random(),random.random()) for i in range(1000,2000)]

def eval(data):

    ##\def gen_data2(x):
    #    return list(zip(x,[math.sin(a) for a in x],[math.cos(a) for a in x]))
    tr1 = LinearRegression()
    t= np.array(data)
    tr1.fit(gen_data1(list(range(0,1000))),t[:,0])
    tr2 = LinearRegression()
    tr2.fit(gen_data1(list(range(0,1000))),t[:,1])


    tr1.coef_

    preds1 = tr1.predict(gen_data1(list(range(1000,2000))))
    preds2 = tr2.predict(gen_data1(list(range(1000,2000))))

    return list(zip(preds1,preds2))
```

Командная часть

Задача II.2.1. (2000 баллов)

Петька и Василий Иванович летят летят самолете, и вдруг Василий Иванович, сидящий за штурвалом пилота, вскрикивает: «Петька, приборы!» Следует ответ: «Тридцать восемь». «Что тридцать восемь?» — «А что приборы?»

В самолете пропали все подписи к приборам, и наши герои не могут определить в облаках летят они в верх или вниз. У героев есть накопленный массив данных, пока был виден горизонт, массив данных представляет из себя значения приборных панелей и целевая переменная бинарная (1 или 0)- летит самолет вниз или вверх. Необходимо спрогнозировать вероятность того, что значение целевой переменной равно 1 (число от 0 до 1) для нового массива значений.

Массив показателей приборов: https://drive.google.com/file/d/17h5dVvTspX92SgVK6hVFe8uPZg0F-Qi_/view?usp=sharing}

Массив значений: https://drive.google.com/file/d/1Nysa9JXfcUD0ubp296pud1Y9zxky4dYx/view?usp=sharing

Необходимо загрузить массив формата zip, где содержится файл submission.py на языке Python, в котором есть функция eval, принимающая на вход массив значений приборов (в томже формате, что файл с показателями) и выгружающая массив вероятностей. Возможно в массив выгрузить модель.

Необходимо сделать архив zip submission.py и model.pkl и загрузить его в качестве решения.

Решение

Обучение

```
import math
import random
import numpy as np
import pandas as pd
import sklearn
from sklearn import datasets
from sklearn import model_selection
from sklearn import metrics
from sklearn import linear_model
import pickle
import lightgbm
import ast
with open('x_train.txt',"r") as f:
    x_train_1  =  np.array(ast.literal_eval(f.read()))
with open('y_train.txt',"r") as f:
    y_train_1  =  np.array(ast.literal_eval(f.read()))
lr = linear_model.LogisticRegression()
lr.fit(x_train_1,y_train_1)
with open("model.pkl","wb") as f:
    pickle.dump(lr,f)
```

Файл submission.py выглядит в этом случае:

```
import random
import numpy as np
import sklearn
import math
import pickle
from sklearn import linear_model
import sys
np.set_printoptions(threshold=sys.maxsize)

def eval(data):
    with open("model.pkl","rb") as f:
        model = pickle.load(f)

    res = model.predict_proba(np.array(data))[:,1]
    return np.array2string(np.array(res),separator=",",precision=20 ).replace("\\n",
    ↪  "\n").replace("\n", " ")
```

Задача II.2.2. (5000 баллов)

Вам дается датасет просмотра российских телеканалов с 2018 года. Аудитория измеряется для различного возраста и различной метрики. (TVR и Share) Про метрики подробно описано здесь http://www.nazaykin.ru/MP/tv/audit_tv.htm.

Поле next_day — техническое, оно нужно, т.к. изначально часы с полуночи до 4х утра относятся к предыдущему дню. Некоторые дни скрыты целиком, необходимо спрогнозировать значения TVR и Share по этим дням по аудитории. Возможно использовать внешние открытые данные, не связанные напрямую с рейтингом передач (прогноз погоды и т. д. но не другие рейтинги).

Тренировочная выборка: `https://drive.google.com/file/d/14C3fWUQk9fuv_BaFFN87ZckAPDDsuFd2/view?usp=sharing`

Необходимо спрогнозировать значения для тестовой выборки, в файле указаны колонки с пустыми значениями: `https://drive.google.com/file/d/1EULSxKCsdmY6XLBbYzXsrX2FBzx1cdCB/view?usp=sharing`

Необходимо в качестве результата вернуть текстовое представление тестовой выборки с заполненными значениями.

Пример решения с генерацией ответа. В качестве решение берется среднее значение за последние 72 часа для данного канала.

Метрика основанная на MSLE: `https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/mean-squared-logarithmic-error-(msle)`.

https://drive.google.com/file/d/1s3drH_GaMrJanDfPaYl0QvfGI5csjAew/view?usp=sharing.

Решение

Обучение

```python
import numpy as np
import sys
import pandas as pd
np.set_printoptions(threshold=sys.maxsize)

def eval(data):
    X_train = pd.read_csv("train.csv")
    X_train["Дата"] = pd.to_datetime(X_train["Дата"])
    X_test = pd.read_csv("test.csv")
    X_test["Дата"] = pd.to_datetime(X_test["Дата"])
    X_all = X_train.append(X_test,sort=True)
    X_all.sort_values(by=["Дата","Канал"],inplace=True)

    X_all.set_index("Дата", inplace=True)
    value_cols = ['Все 18+_TVR', 'Все 55+_TVR','Все 18+_Share', 'Все 55+_Share']
    chanels = X_test["Канал"].unique().tolist()
    for n in value_cols:
        #print(n,X_all[n].isna().sum())
        for c in chanels:
            X_all.loc[X_all["Канал"]==c,n] = X_all.loc[X_all["Канал"]==c,n].rolling(
                                                73,min_periods=1).mean()
        #print("rolls",X_all[n].isna().sum())
```

```
23
24      X_res_2 = X_all.loc[X_all["Дата_День"].isin( X_test["Дата_День"]
        ↪  )].reset_index().sort_values(by=["Дата", "Канал"])
25      str_x_res = X_res_2.to_csv()
26
27      return str_x_res
```