

ДОДАТОК А

Текст програми

```
import numpy as np
import pandas as pd
from google.colab import files

uploaded = files.upload()
import io
df = pd.read_csv(io.StringIO(uploaded['RAW_recipes.csv'].decode('utf-8')))
import ast
dataset1 = df.tags
tags = []
for x in dataset1:
    for el in ast.literal_eval(x):
        if el in tags:
            continue
        else:
            tags.append(el)
unique_tags = list(set(tags))
ingredients = [];
for x in df.ingredients:
    for el in ast.literal_eval(x):
        if el not in ingredients:
            ingredients.append(el)
ing1 = [];
for el in ingredients:
    el_arr = el.split(' ');
    for y in el_arr:
        ing1.append(y)
unique_ing = list(set(ing1))
```

```

final_tags = []
for i in tags:
    if i not in unique_ing:
        final_tags.append(i)
len(final_tags)
df1 = pd.DataFrame(final_tags, columns=['tag'])
df[['calories', 'total fat (PDV)', 'sugar (PDV)', 'sodium (PDV)', 'protein
(PDV)', 'saturated fat (PDV)', 'carbohydrates (PDV)']] = df.nutrition.s
tr.split(",", expand=True)
df['calories'] = df['calories'].apply(lambda x: x.replace('[', ''))
df['carbohydrates (PDV)'] = df['carbohydrates (PDV)'].apply(lambda x:
x.replace(']', ''))
df[['calories', 'total fat (PDV)', 'sugar (PDV)', 'sodium (PDV)', 'protein
(PDV)', 'saturated fat (PDV)', 'carbohydrates (PDV)']] = df[['calories'
, 'total fat (PDV)', 'sugar (PDV)', 'sodium (PDV)', 'protein (PDV)', 'satur
ated fat (PDV)', 'carbohydrates (PDV)']].astype('float')
df.drop(['id', 'contributor_id', 'submitted', 'nutrition'], axis=1, inpla
ce = True)
df['Lactose'] = np.nan
df['Lactose'] = df['Lactose'].astype('bool')
for i in df['tags'].index:
    if 'lactose' in df['tags'][i]:
        df['Lactose'][i]=True
    else:
        df['Lactose'][i]=False
df['Easy'] = np.nan
df['Easy'] = df['Easy'].astype('bool')
for i in df['tags'].index:
    if 'easy' in df['tags'][i]:
        df['Easy'][i]=False
    else:
        df['Easy'][i]=True
df['Nuts'] = np.nan

```

```

df['Nuts'] = df['Nuts'].astype('bool')
for i in df['tags'].index:
    if 'nut-free' in df['tags'][i]:
        df['Nuts'][i]=False
    else:
        df['Nuts'][i]=True
df['Dietary'] = np.nan
df['Dietary'] = df['Dietary'].astype('bool')
for i in df['tags'].index:
    if 'dietary' in df['tags'][i]:
        df['Dietary'][i]=True
    else:
        df['Dietary'][i]=False
df['Diabetic'] = np.nan
df['Diabetic'] = df['Diabetic'].astype('bool')
for i in df['tags'].index:
    if 'diabetic' in df['tags'][i]:
        df['Diabetic'][i]=True
    else:
        df['Diabetic'][i]=False
df['Kids'] = np.nan
df['Kids'] = df['Kids'].astype('bool')
kids_tags = ['kid-friendly', 'toddler-friendly', 'infant-baby-
friendly', 'snacks-kid-friendly']
for i in df['tags'].index:
    kids_count = 0;
    for j in range(0, len(kids_tags)):
        if kids_tags[j] in df['tags'][i]:
            kids_count = kids_count + 1;
            df['Kids'][i]=True
    if kids_count == 0:
        df['Kids'][i]=False
df['Dish'] = np.nan

```

```

dish_tags = []
arr = ['main-dish', 'side-
dish', 'salad', 'soup', 'dessert', 'brunch', 'lunch', 'quick-
breads', 'garnish']
for tag in final_tags:
    for x in arr:
        if x in tag:
            dish_tags.append(tag)
dish_dtype = pd.api.types.CategoricalDtype(
    categories=dish_tags)
df['Dish'] = df['Dish'].astype(dish_dtype)
for i in df['tags'].index:
    for j in range(0, len(dish_tags)):
        if dish_tags[j] in df['tags'][i]:
            df['Dish'][i]=dish_tags[j]
df['Quisine'] = np.nan
quisine_tags = []
arr = ['mongolian', 'laotian', 'namibian', 'venezuelan', 'cambodian', 'beij
ing', 'irish', 'angolan', 'austrian', 'vietnamese',
    'palestinian', 'nepalese', 'icelandic', 'malaysian', 'chilean', 'peruvian'
, 'egyptian', 'argentine', 'sudanese', 'congolese',
    'ecuadorean', 'brazilian', 'nigerian', 'ethiopian', 'georgian', 'jewish',
    'iranian', 'costa-rican', 'guatemalan', 'honduran',
    'finnish', 'czech', 'filipino', 'colombian', 'chinese', 'dutch', 'south-
african', 'new-zealand', 'pakistani', 'scottish',
    'chutneys', 'somalian', 'iraqi', 'saudi-
arabian', 'polynesian', 'puerto-rican', 'native-american', 'south-
american',
    'middle-eastern', 'central-american', 'scandinavian', 'pacific-
northwest', 'midwestern', 'southern-united-states',
    'californian', 'south-west-pacific', 'north-american']
for tag in final_tags:
    for x in arr:

```

```

    if x in tag:
        quisine_tags.append(tag)
quisine_dtype = pd.api.types.CategoricalDtype(
    categories=quisine_tags)
df['Quisine'] = df['Quisine'].astype(quisine_dtype)
for i in df['tags'].index:
    for j in range(0, len(quisine_tags)):
        if quisine_tags[j] in df['tags'][i]:
            df['Quisine'][i]=quisine_tags[j]
df['Groups'] = np.nan
df['Groups'] = df['Groups'].astype('bool')
for i in df['tags'].index:
    if 'for-large-groups' in df['tags'][i]:
        df['Groups'][i]=True
    else:
        df['Groups'][i]=False
df['Vegan'] = np.nan
veg_tags = []
arr = ['bird', 'pork', 'chicken', 'ham', 'lamb', 'meat', 'eggs', 'beef']
for tag in final_tags:
    for x in arr:
        if x in tag:
            veg_tags.append(tag)
df['Vegan'] = df['Vegan'].astype('bool')
for i in df['tags'].index:
    veg_count = 0;
    for j in range(len(veg_tags)):
        if veg_tags[j] not in df['tags'][i]:
            veg_count = veg_count + 1;
    if veg_count == len(veg_tags):
        df['Vegan'][i]=True
    else:
        df['Vegan'][i]=False

```

```

# result = df.iloc[0:100];
result = df;
warning = False;
pd.options.mode.chained_assignment = None
def countSimilarity(df, target_id):
    global result;
    result['Similarity'] = np.nan;
    targ = df.loc[[target_id]];
    for i in df.index:
        steps_s = 1 - (abs(df.n_steps[i] - targ.n_steps) / (max(df.n_steps
) - min(df.n_steps)));
        minutes_s = 1 - (abs(df.minutes[i] - targ.minutes) / (max(df.minut
es) - min(df.minutes)));
        n INGR_s = 1 - (abs(df.n_ingredients[i] - targ.n_ingredients) / (ma
x(df.n_ingredients) - min(df.n_ingredients)));
        calories_s = 1 - (abs(df.calories[i] - targ.calories) / (max(df.ca
lories) - min(df.calories)));
        tfat_s = 1 - (abs(df['total fat (PDV)'][i] - targ['total fat (PDV)
']) / (max(df['total fat (PDV)']) - min(df['total fat (PDV)'])));
        sugar_s = 1 - (abs(df['sugar (PDV)'][i] - targ['sugar (PDV)']) / (
max(df['sugar (PDV)']) - min(df['sugar (PDV)'])));
        sodium_s = 1 - (abs(df['sodium (PDV)'][i] - targ['sodium (PDV)'])
/ (max(df['sodium (PDV)']) - min(df['sodium (PDV)'])));
        protein_s = 1 - (abs(df['protein (PDV)'][i] - targ['protein (PDV)
']) / (max(df['protein (PDV)']) - min(df['protein (PDV)'])));
        sfat_s = 1 - (abs(df['saturated fat (PDV)'][i] - targ['saturated f
at (PDV)']) / (max(df['saturated fat (PDV)']) - min(df['saturated fat
(PDV)'])));
        carbs_s = 1 - (abs(df['carbohydrates (PDV)'][i] - targ['carbohydra
tes (PDV)']) / (max(df['carbohydrates (PDV)']) - min(df['carbohydrates
(PDV)'])));
        dish_s = 1 if pd.isna(df.Dish[i]) and pd.isna(targ.iloc[0].Dish) o
r df.Dish[i] == targ.iloc[0].Dish else 0;

```

```

    quisine_s = 1 if pd.isna(df.Quisine[i]) and pd.isna(targ.iloc[0].Q
uisine) or df.Quisine[i] == targ.iloc[0].Quisine else 0;
    group_s = 1 if pd.isna(df.Groups[i]) and pd.isna(targ.iloc[0].Gro
ups) or df.Groups[i] == targ.iloc[0].Groups else 0;
    tags_df = pd.DataFrame({'tags': ast.literal_eval(targ.iloc[0].tags
)}));
    z = tags_df.isin(ast.literal_eval(df.tags[i]));
    tags_s = len(z[z.tags==True])/len(z);
    ingr_df = pd.DataFrame({'ingredients': ast.literal_eval(targ.iloc[
0].ingredients)});
    z = ingr_df.isin(ast.literal_eval(df.ingredients[i]));
    ingr_s = len(z[z.ingredients==True])/len(z);

```

```

    sim = 0.3*steps_s[[target_id]] + 0.2*minutes_s[[target_id]] + 0.2*
ningr_s[[target_id]] + 0.7*calories_s[[target_id]] + 0.1*tfat_s[[targe
t_id]] + 0.1*sugar_s[[target_id]] + 0.1*sodium_s[[target_id]] + 0.1*pr
otein_s[[target_id]] + 0.1*sfat_s[[target_id]] + 0.1*carbs_s[[target_i
d]] + 0.5*dish_s + 0.4*quisine_s + group_s + 0.8*tags_s + 0.9*ingr_s

```

```

    result['Similarity'][i] = sim/5.6;
    return result

```

```

def checkResult(len, category):

```

```

    global warning;

```

```

    if len < 20 and warning == False:

```

```

        print('Change your recipe restrictions if possible, reccomendation
s amount is small.');
```

```

        print('We recommend changing the ' + category + ' restriction.');
```

```

        warning = True;

```

```

def recommend(user, recipe, target_id):

```

```

global warning;
if target_id == False:
    global result;
    print('Initial', len(result));
    if user['age'] <= 6:
        result = result[result['Kids']]
        print('age!', len(result));
    if len(user['allergies']) != 0:
        allerg = pd.DataFrame({'ingredients': user['allergies']});
        allerg_indices = [];
        for z in range(0, len(result)):
            b = any(allerg.isin(ast.literal_eval(result['ingredients'][z])
)['ingredients'])
                if b:
                    allerg_indices.append(z)
            result = result[result.index.isin(allerg_indices)];
            print('allergies!', len(result))
    if len(user['issues']) != 0:
        if 'Lactose' in user['issues']:
            result = result[result['Lactose']==False]
        if 'Nuts' in user['issues']:
            result = result[result['Nuts']==False]
        if 'Diabetic' in user['issues']:
            result = result[result['Diabetic']]
        if 'Vegan' in user['issues']:
            result = result[result['Vegan']]
        print('issues!', len(result))
    if user['calories']:
        result = result[result['calories']<=user['calories']]
        print('calories!', len(result))
    if user['diet']:
        result = result[result['Dietary']]
        print('diet!', len(result));

```

```

if user['advanced_diet']:
    print('advanced', len(result));
    if user['advanced_diet']['total_fat']:
        result = result[result['total fat (PDV)']<=user['advanced_diet
']['total_fat']]
    if user['advanced_diet']['sugar']:
        result = result[result['sugar (PDV)']<=user['advanced_diet']['
sugar']]
    if user['advanced_diet']['sodium']:
        result = result[result['sodium (PDV)']<=user['advanced_diet']['
'sodium']]
    if user['advanced_diet']['protein']:
        result = result[result['protein (PDV)']<=user['advanced_diet']
['protein']]
    if user['advanced_diet']['saturated_fat']:
        result = result[result['saturated fat (PDV)']<=user['advanced_
diet']['saturated_fat']]
    if user['skill']:
        print('skill!', len(result));
        if user['skill'] == 'beginner':
            result = result[result['Easy']]

if len(result) < 20:
    print('Change your user restrictions if possible, reccomendation
s amount is small. ');
    warning = True;

res = result;

if recipe['dish_complexity']:
    if recipe['dish_complexity'] == 'Easy':
        res = res[res['n_steps'] < 8]
    if recipe['dish_complexity'] == 'Medium':

```

```

    res = res[res['n_steps'] <= 15]
else:
    res = res[res['n_steps'] > 15]
checkResult(len(res), 'Dish complexity');
print('complexity!', len(res));
if recipe['calories']:
    res = res[res['calories']<=recipe['calories']]
    checkResult(len(res), 'Dish calories');
    print('calories!', len(res));
if recipe['advanced_diet']:
    if recipe['advanced_diet']['total_fat']:
        res = res[res['total fat (PDV)']<=recipe['advanced_diet']['total_fat']]
    if recipe['advanced_diet']['sugar']:
        res = res[res['sugar (PDV)']<=recipe['advanced_diet']['sugar']]
    if recipe['advanced_diet']['sodium']:
        res = res[res['sodium (PDV)']<=recipe['advanced_diet']['sodium']]
    if recipe['advanced_diet']['protein']:
        res = res[res['protein (PDV)']<=recipe['advanced_diet']['protein']]
    if recipe['advanced_diet']['saturated_fat']:
        res = res[res['saturated fat (PDV)']<=recipe['advanced_diet']['saturated_fat']]
    checkResult(len(res), 'Dish nutrition');
    print('advanced!', len(res));
if recipe['time_to_make']:
    res = res[res['minutes']<=recipe['time_to_make']]
    checkResult(len(res), 'Time to make');
    print('time!', len(res));
if recipe['for_groups']:
    res = res[res['Groups']]

```

```

    checkResult(len(res), 'Number of servings');
    print('groups!', len(res));
if recipe['cuisine']:
    res = res[res['Cuisine'] == recipe['cuisine']]
    checkResult(len(res), 'Cuisine');
    print('cuisine!', len(res));
if recipe['dish']:
    res = res[res['Dish'] == recipe['dish']]
    checkResult(len(res), 'Dish type')
    print('dish!', len(res));
if len(recipe['ingredients']) != 0:
    print('dish1!', len(res));
    ingr = pd.DataFrame({'ingredients': recipe['ingredients']});
    ingr_indices = [];
    for i,row in res.iterrows():
        b = all(ingr.isin(ast.literal_eval(res['ingredients'][i]))['in
redients'])
        if b:
            ingr_indices.append(i)
    res = res[res.index.isin(ingr_indices)];
    checkResult(len(res), 'Ingredients');
    print('ingr!', len(res));
if len(recipe['tags']) != 0:
    tag = pd.DataFrame({'tags': recipe['tags']});
    tags_indices = [];
    for i,row in res.iterrows():
        b = all(tag.isin(ast.literal_eval(res['tags'][i]))['tags'])
        if b:
            tags_indices.append(i)
    res = res[res.index.isin(tags_indices)];
    checkResult(len(res), 'Additional properties');
    print('tags!', len(res));
print('final', len(res));

```

```
    return res;
else:
    countSimilarity(result, target_id);
return result
query = { "user": {
    'age': 18,
    'allergies': ['pork','eggs'],
    'issues': ['Diabetic'],
    'calories': 1500,
    'diet': True,
    'advanced_diet': False,
    'skill': False
}, "recipe":{
    'dish_complexity': 'Medium',
    'calories': 1500,
    'advanced_diet': False,
    'time_to_make': 170,
    'for_groups': False,
    'quisine': False,
    'dish': False,
    'ingredients': ['potato','salt','pepper'],
    'tags':[]
}, "target": False}
recommend(query["user"],query["recipe"],query["target"])
countSimilarity(result,3122)
len(result)
rs = result.sort_values(by='Similarity')
rs.plot(x='name',y="Similarity")
```

