КПІ ім. Ігоря Сікорського

Інститут прикладного системного аналізу

Кафедра системного проектування

Розрахунково-графічна робота

з курсу: «Машинне навчання»

на тему: «Прогнозування категорії страв за інгредієнтами»

Виконала:

Студентка групи ДА-81

ННК «ІПСА»

Козінцева Анна Олексіївна

Київ – 2021

**Мета роботи:** навчитися використовувати засоби для візуалізації даних та прогнозування результату.

**Завдання:**

Побудувати модель для прогнозу даних з Kaggle-змагання за посиланням:

https://www.kaggle.com/c/whats-cooking

Спрогнозувати за наявними у датасеті інгредієнтами кухню країни, до якої вони відносяться.

**Опис даних:**

**train.json.zip** – датасет для тренування, який містить id рецепту, тип національної кухні та список інгредієнтів для кожного рецепту.

**test.json.zip** – датасет для тестування, який містить id рецепту та список інгредієнтів для кожного рецепту.

У файлі для тестування test.json формат рецепту такий же, як у train.json, але відсутній тип національної кухні, адже він є target-ом, який намагаємося спрогнозувати.

*Приклад рецепту з train.json:*

{  
 "id": 24717,  
 "cuisine": "indian",  
 "ingredients": [  
 "tumeric",  
 "vegetable stock",  
 "tomatoes",  
 "garam masala",  
 "naan",  
 "red lentils",  
 "red chili peppers",  
 "onions",  
 "spinach",  
 "sweet potatoes"  
 ]  
 },

**Оформлення результату:**

Файл прогнозів повинен містити в собі id рецепту та відповідний тип національної кухні, і бути оформлений наступним чином:

id,cuisine  
35203,italian  
17600,italian  
35200,italian  
17602,italian  
...  
etc.

**Короткі теоретичні відомості**

**EDA (exploratory data analysis) –** «розвідковий аналіз даних» – попередній аналіз даних з метою виявлення найзагальніших закономірностей та тенденцій, характеру та властивостей даних аналізу, законів розподілу величин, які аналізуються.

*Основні цілі розвідкового аналізу:*

* максимальне «проникнення» в дані;
* виявлення основних структур;
* вибір найвагоміших змінних;
* виявлення відхилень та аномалій;
* перевірка основних гіпотез (припущень);
* розробка початкових моделей.

**Data Preprocessing** – передобробка даних, яка виконується для підготовки даних до аналізу. В результаті цієї обробки дані приводяться у відповідності до вимог, які визначаються специфікою конкретної задачі.

Передобробка даних включає два напрямки: очистку та оптимізацію.

* **Очистка** – відбувається з метою виключення різного роду факторів, які знижують якість даних та заважають роботі аналітичних алгоритмів. Вона включає обробку дублікатів, суперечностей, відновлення та заповнення пропусків, перетворення некоректних форматів.
* **Оптимізація** – включає зменшення розмірності, виявлення та виключення неважливих ознак.

Основна відмінність оптимізації від очистки полягає в тому, що фактори, які усуваються в процесі очистки, суттєво знижують точність розв’язку задачі або роблять роботу аналітичних алгоритмів неможливою. Проблеми, які вирішуються при оптимізації, адаптують дані до конкретної задачі та підвищують ефективність їхнього аналізу.

**Моделювання** – процес побудови алгоритму для отримання прогнозів.

**Хід роботи**

1. Після зчитування даних відбувається EDA (exploratory data analusis) для графічного представлення та кращого розуміння даних

Наприклад, будуємо гістограму, де показано

* страв якої країни у тестовому датасеті найбільше:

fig, ax = plt.subplots(figsize=(10,10))

per\_vals = round(df["cuisine"].value\_counts(normalize=True)\*100, 2)

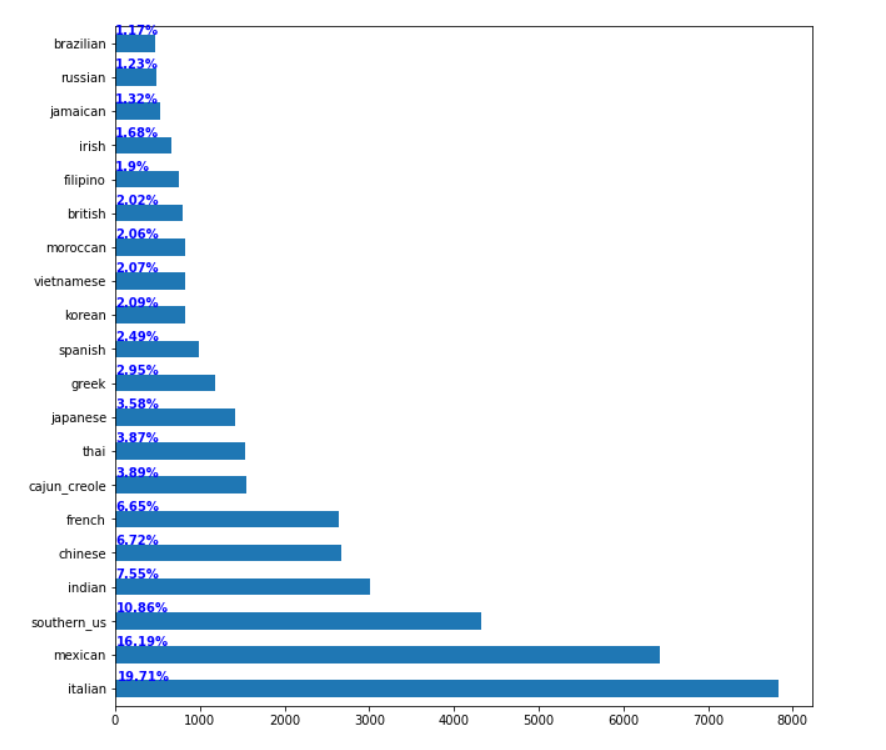
for i, v **in** enumerate(per\_vals):

ax.text(v + 3, i + .25, str(v)+"%", color='blue', fontweight='bold')

df["cuisine"].value\_counts().plot.barh(ax = ax)

plt.show()

Отримали результат:



Бачимо, що італійських страв у даному датасеті найбільше.

* які інгредієнти найчастіше застосовуються

fig, ax = plt.subplots(figsize=(22,7))

extensive\_ing\_list = []

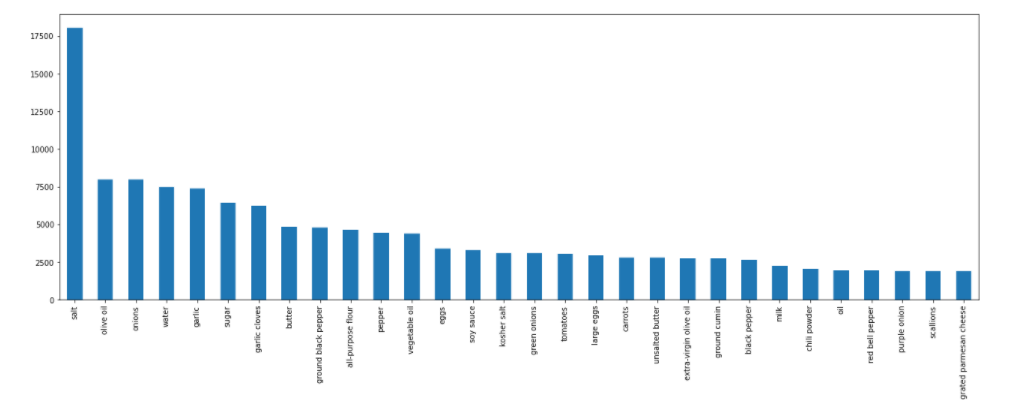
for x **in** df['ingredients']:

for y **in** x:

extensive\_ing\_list.append(y)

extensive\_ing\_list = pd.Series(extensive\_ing\_list)

extensive\_ing\_list.value\_counts().sort\_values(ascending=False).head(30).plot.bar(ax = ax)



До ТОП-5 входять: сіль, оливкова олія, цибуля, вода та часник.

Аналогічно знаходимо інгредієнти, які найчастіше застосовуються у кухні кожної країни.

**2. Попередня обробка даних**

У даному датасеті інгредієнти представлені у вигляді списку. Для зручнішої роботи перетворюємо їх у рядковий тип.

*Функції для обробки:*

def preprocess\_df(df):

def process\_string(x):

x = [" ".join([WordNetLemmatizer().lemmatize(q) for q **in** p.split()]) for p **in** x] *#Lemmatization*

x = list(map(lambda x: re.sub(r'\(.\*oz.\)|crushed|crumbles|ground|minced|powder|chopped|sliced','', x), x))

x = list(map(lambda x: re.sub("[^a-zA-Z]", " ", x), x)) *# прибираємо все окрім a-z та A-Z*

x = " ".join(x) *# перетворюємо list-елементи у string-елементи*

x = x.lower()

return x

df = df.drop('id',axis=1)

df['ingredients'] = df['ingredients'].apply(process\_string)

return df

та

def get\_cuisine\_cumulated\_ingredients(df):

cuisine\_df = pd.DataFrame(columns=['ingredients'])

for cus **in** cuisine:

st = ""

for x **in** df[df.cuisine == cus]['ingredients']:

st += x

st += " "

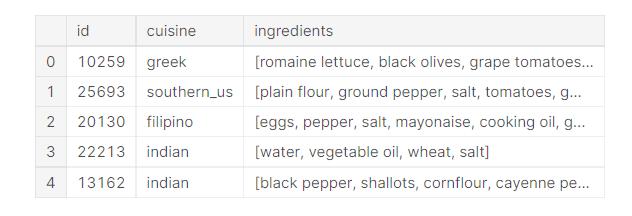
cuisine\_df.loc[cus,'ingredients'] = st

cuisine\_df = cuisine\_df.reset\_index()

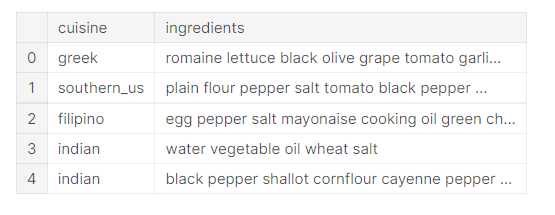
cuisine\_df = cuisine\_df.rename(columns ={'index':'cuisine'})

return cuisine\_df

*Початкові дані:*



*Дані після обробки:*

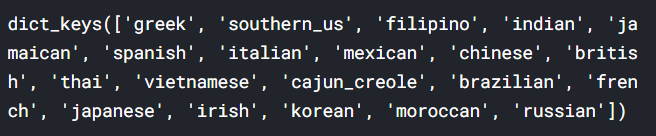


**3. Кластеризація**

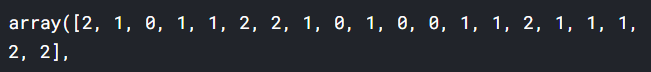
Виконуємо для того, щоб знайти щось спільне між деякими кухнями.

Для цього використаємо алгоритм KMeans.

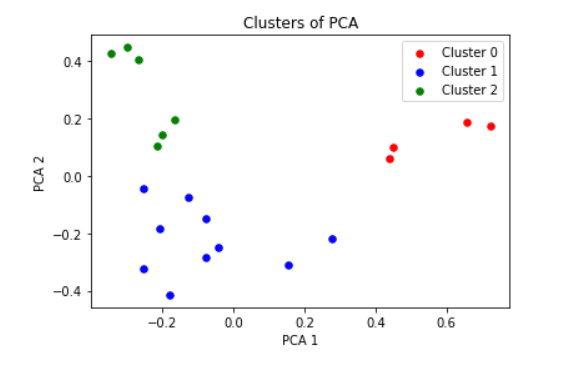
Для наступних 20-ти країн:



отримуємо їхні розподілення за відповідними кластерами (за відповідним індексом):



Графічне представлення:



**4. Моделювання**

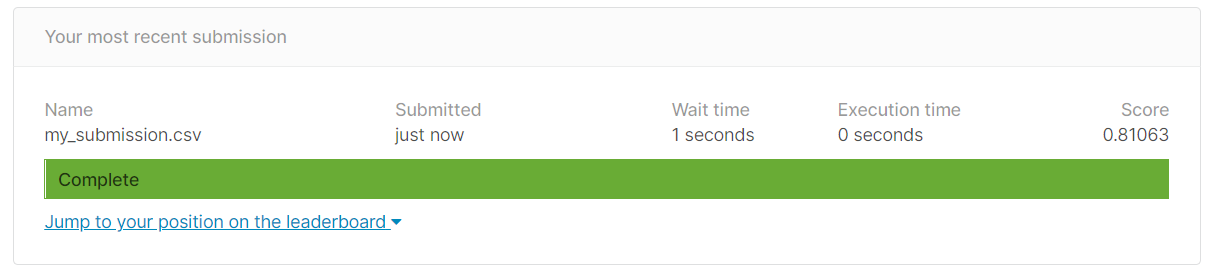
Для моделювання використаємо алгоритм Linear SVC.

Метод опорних векторів (SVM – Supported vector machines) – це модель (алгоритм), який використовується для пошуку площини, яка розбиває простір вибірок.

Вона може застосовуватися для класифікації (SVC) та для регресії (SVR). У даній роботі – для класифікації.

Нижче наведено лістинг до програми.

**Результат:**



Отримали score = 0.81063

**Лістинг**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import sklearn

import os

import json

import re

import nltk

import zipfile

from datetime import datetime

from sklearn.preprocessing import LabelEncoder

from nltk.stem import WordNetLemmatizer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.metrics import confusion\_matrix, accuracy\_score

from sklearn.feature\_extraction.text import CountVectorizer

Зчитуємо дані

for t **in** ['train','test']:

with zipfile.ZipFile("../input/whats-cooking/**{}**.json.zip".format(t),"r") as z:

z.extractall(".")

with open('./train.json') as data\_file:

data = json.load(data\_file)

with open('./test.json') as test\_file:

test = json.load(test\_file)

Приклад train-датасету

df = pd.DataFrame(data)

test\_df = pd.DataFrame(test)

test\_ids = test\_df['id']

df.head()

EDA

(df.isnull().sum() / len(df))\*100 *# жодних відсутніх даних у train*

(test\_df.isnull().sum() / len(test\_df))\*100 *# жодних відсутніх даних у test*

Страв якої країни найбільше

fig, ax = plt.subplots(figsize=(10,10))

per\_vals = round(df["cuisine"].value\_counts(normalize=True)\*100, 2)

for i, v **in** enumerate(per\_vals):

ax.text(v + 3, i + .25, str(v)+"%", color='blue', fontweight='bold')

df["cuisine"].value\_counts().plot.barh(ax = ax)

plt.show()

Які інгредієнти найчастіше застосовуються

fig, ax = plt.subplots(figsize=(22,7))

extensive\_ing\_list = []

for x **in** df['ingredients']:

for y **in** x:

extensive\_ing\_list.append(y)

extensive\_ing\_list = pd.Series(extensive\_ing\_list)

extensive\_ing\_list.value\_counts().sort\_values(ascending=False).head(30).plot.bar(ax = ax)

Перелік усіх cuisine

cuisine = df["cuisine"].unique()

all\_cus = dict()

for cs **in** cuisine:

i = []

for ing\_list **in** df[df['cuisine']==cs]['ingredients']:

for ing **in** ing\_list:

i.append(ing)

all\_cus[cs] = i

all\_cus.keys()

25 найчастіше застосовуваних інгредієнтів для кухні кожної країни

for key **in** all\_cus.keys():

fig, ax = plt.subplots(figsize=(25,2))

pd.Series(all\_cus[key]).value\_counts().head(25).plot.bar(ax=ax, title=key)

plt.show()

Попередня обробка даних

def preprocess\_df(df):

def process\_string(x):

x = [" ".join([WordNetLemmatizer().lemmatize(q) for q **in** p.split()]) for p **in** x] *#Lemmatization*

x = list(map(lambda x: re.sub(r'\(.\*oz.\)|crushed|crumbles|ground|minced|powder|chopped|sliced','', x), x))

x = list(map(lambda x: re.sub("[^a-zA-Z]", " ", x), x)) *# прибираємо все окрім a-z та A-Z*

x = " ".join(x) *# перетворюємо list-елементи у string-елементи*

x = x.lower()

return x

df = df.drop('id',axis=1)

df['ingredients'] = df['ingredients'].apply(process\_string)

return df

def get\_cuisine\_cumulated\_ingredients(df):

cuisine\_df = pd.DataFrame(columns=['ingredients'])

for cus **in** cuisine:

st = ""

for x **in** df[df.cuisine == cus]['ingredients']:

st += x

st += " "

cuisine\_df.loc[cus,'ingredients'] = st

cuisine\_df = cuisine\_df.reset\_index()

cuisine\_df = cuisine\_df.rename(columns ={'index':'cuisine'})

return cuisine\_df

df = preprocess\_df(df)

test\_df = preprocess\_df(test\_df)

cuisine\_df = get\_cuisine\_cumulated\_ingredients(df)

df.head()

train = df['ingredients']

target = df['cuisine']

test = test\_df['ingredients']

Count Vectorizer

def count\_vectorizer(train, test=None):

cv = CountVectorizer()

train = cv.fit\_transform(train)

if test **is** **not** None:

test = cv.transform(test)

return train, test, cv

else:

return train, cv

TFiDF Vectorizer

def tfidf\_vectorizer(train, test=None):

tfidf = TfidfVectorizer(stop\_words='english',

ngram\_range = ( 1 , 1 ),analyzer="word",

max\_df = .57 , binary=False , token\_pattern=r'\w+' , sublinear\_tf=False)

train = tfidf.fit\_transform(train)

if test **is** **not** None:

test = tfidf.transform(test)

return train, test, tfidf

else:

return train, tfidf

train\_tfidf, test\_tfidf, tfidf = tfidf\_vectorizer(train,test)

cuisine\_data\_tfidf, cuisine\_tfidf = tfidf\_vectorizer(cuisine\_df['ingredients'])

Кластеризація

from sklearn.cluster import KMeans

from sklearn.decomposition import KernelPCA,PCA,TruncatedSVD

def get\_kmeans\_wcss(data, n\_limit=15):

wcss = [] *#Within cluster sum of squares (WCSS)*

for i **in** range(1,n\_limit):

km = KMeans(init='k-means++', n\_clusters=i, n\_init=10)

km.fit(data)

wcss.append(km.inertia\_)

plt.title("Elbow Method")

plt.plot(range(1, n\_limit), wcss)

plt.xlabel("Number of clusters")

plt.ylabel("WCSS")

return wcss

def kmeans(data, n):

km = KMeans(init='k-means++', n\_clusters=n, n\_init=10)

km = km.fit(data)

return km.predict(data), km

def get\_PCA(data, n\_components=2):

pca = PCA(n\_components = n\_components)

reduced\_data = pca.fit\_transform(data)

explained\_variance = pca.explained\_variance\_ratio\_

print(explained\_variance)

return reduced\_data, pca, explained\_variance

def get\_kernel\_PCA(data, n\_components=2, kernel='rbf'):

kpca = KernelPCA(n\_components = 2, kernel = kernel)

reduced\_data = kpca.fit\_transform(data)

explained\_variance = kpca.explained\_variance\_ratio\_

print(explained\_variance)

return reduced\_data, kpca, explained\_variance

def get\_TSVD(data, n\_components=2, n\_ittr=5, algorithm='randomized'):

tsvd = TruncatedSVD(n\_components=n\_components, n\_iter=n\_ittr, algorithm=algorithm)

reduced\_data = tsvd.fit\_transform(data)

explained\_variance = tsvd.explained\_variance\_ratio\_

print(explained\_variance)

return reduced\_data, tsvd, explained\_variance

def create\_pca\_graph(cluster\_pca, red\_pca, n\_clus):

c\_mask = []

c\_x = []

c\_y = []

for i **in** range(0,n\_clus):

c\_mask.append([x for x **in** cluster\_pca==i])

for i **in** range(0,n\_clus):

c\_x.append([a[0] for a, b **in** zip(red\_pca, c\_mask[i]) if b])

c\_y.append([a[1] for a, b **in** zip(red\_pca, c\_mask[i]) if b])

colours = ['red','blue','green','orange','purple','cyan','black','magenta']

for i **in** range(0,n\_clus):

plt.scatter(c\_x[i], c\_y[i], s=30, c=colours[i], label='Cluster **{}**'.format(i))

plt.title("Clusters of PCA")

plt.xlabel("PCA 1")

plt.ylabel("PCA 2")

plt.legend()

plt.show()

red\_cuisine\_pca, cus\_pca, var\_cus\_pca = get\_PCA((cuisine\_data\_tfidf).toarray(),2)

wcss\_pca = get\_kmeans\_wcss(red\_cuisine\_pca,20)

cluster\_cus\_pca, km\_cus\_pca = kmeans(red\_cuisine\_pca,3)

cluster\_cus\_pca

create\_pca\_graph(cluster\_cus\_pca, red\_cuisine\_pca, 3)

Моделювання

from scipy import sparse

*# для кращих результатів*

train = train\_tfidf

test = test\_tfidf

Linear SVC

from sklearn.svm import LinearSVC, SVC

from sklearn.metrics import f1\_score

param\_grid = {'C': [0.001, 0.1, 1, 10, 50, 100, 500, 1000, 5000],

'penalty': ['l1','l2'],

'loss': ['hinge','squared hinge']}

grid = GridSearchCV(LinearSVC(), param\_grid, refit = True, verbose = 3, n\_jobs=-1, scoring='f1\_micro')

grid.fit(train, target)

grid.best\_params\_

grid.best\_score\_

from sklearn.metrics import f1\_score

from sklearn.svm import LinearSVC, SVC

def evalfn(C, gamma):

s = SVC(C=float(C), gamma=float(gamma), kernel='rbf', class\_weight='balanced')

f = cross\_val\_score(s, train, target, cv=5, scoring='f1\_micro')

return f.max()

from bayes\_opt import BayesianOptimization

from sklearn.model\_selection import cross\_val\_score

new\_opt = BayesianOptimization(evalfn, {'C': (0.1, 1000),

'gamma': (0.0001, 1) })

C = 604.5300203551828

gamma = 0.9656489284085462

clf = SVC(C=float(C), gamma=float(gamma), kernel='rbf')

clf.fit(train, target)

y\_pred = clf.predict(test)

my\_submission = pd.DataFrame({'id':test\_ids})

my\_submission['cuisine'] = y\_pred

my\_submission.to\_csv('submission\_**{}**.csv', index=False)

print('Saved file to disk as submission\_**{}**.csv.')

Детальніший лістинг – за посиланням: https://www.kaggle.com/annakozintseva/ml-rgr

**Висновок:** У даній роботі було досліджено датасет, який складається з id рецепту та списку інгредієнтів до цього рецепту. Target – кухня країни, до якої він відноситься. Було проведено EDA-аналіз, що дало детальніше розуміння даних з датасету. Далі дані були приведені до рядкового типу для зручнішої роботи з ними. На основі алгоритму KMeans було виділено кластери для cuisine, а використовуючи алгоритм Linear SVC – побудовано модель для прогнозу та перевірено результат на test-датасеті, в результаті чого було отримано score = 0.81063.