Лабораторное занятие 11. Обработка данных (категориальных данных), характеризующих общие свойства и связи конкретных явлений.

11-зертханалық сабақ. Нақтылы құбылыстардың жалпы қасиеттері мен байланыстарын сипаттайтын деректерді (категориялық деректерді) өңдеу.

- feature extraction and feature engineering – превращение данных, специфических для предметной области, в понятные для модели векторы;
  - feature transformation – трансформация данных для повышения точности алгоритма;
  - feature selection – отсечение ненужных признаков

Пример

Описание данных

В этом примере, вы будете предсказывать, насколько популярны список аренды квартиры на основе содержания листинга, как описание текста, фотографии, количество спален, цена и т.д. Данные поступают из renthop.com,веб-сайт листинг апартизм квартиры. Эти апартаменты расположены в Нью-йорке.

Целевая переменная, interest_level,определяется количеством запросов, которые листинг имеет в продолжительности, что объявление было жить на сайте.

Описания файлов

- train.json - тренировочный набор
- test.json - the test set
- sample_submission.csv - a sample submission file in the correct format
- images_sample.zip - listing images organized by listing_id (a sample of 100 listings)
- Kaggle-renthop.7z - (optional) listing images organized by listing_id. Total size: 78.5GB compressed. Distributed by BitTorrent (Kaggle-renthop.torrent).

Data fields

- bathrooms: number of bathrooms
- bedrooms: number of bathrooms
- building_id
- created
- description
- display_address
- features: a list of features about this apartment
- latitude

- listing_id
- longitude
- manager_id
- photos: a list of photo links. You are welcome to download the pictures yourselves from renthop's site, but they are the same as imgs.zip.
- price: in USD
- street_address
- interest_level: this is the target variable. It has 3 categories: 'high', 'medium', 'low'

```
# перед началом работы не забудьте скачать файл train.json.zip с Kaggle и разархивировать его
(https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries/rules)
import json
import pandas as pd

# сразу загрузим датасет от Renthop
with open('train.json', 'r') as raw_data:
    data = json.load(raw_data)
    df = pd.DataFrame(data)

from functools import reduce
import numpy as np

texts = [['i', 'have', 'a', 'cat'],
         ['he', 'have', 'a', 'dog'],
         ['he', 'and', 'i', 'have', 'a', 'cat', 'and', 'a', 'dog']]

dictionary = list(enumerate(set(reduce(lambda x, y: x + y, texts))))

def vectorize(text):
    vector = np.zeros(len(dictionary))
    for i, word in dictionary:
        num = 0
        for w in text:
            if w == word:
                num += 1
        if num:
            vector[i] = num
    return vector
```

```
for t in texts:
    print(vectorize(t))
```

```
In : from sklearn.feature_extraction.text import CountVectorizer

In : vect = CountVectorizer(ngram_range=(1,1))

In : vect.fit_transform(['no i have cows', 'i have no cows']).toarray()
Out:
array([[1, 1, 1],
       [1, 1, 1]], dtype=int64)

In : vect.vocabulary_
Out: {'cows': 0, 'have': 1, 'no': 2}

In : vect = CountVectorizer(ngram_range=(1,2))

In : vect.fit_transform(['no i have cows', 'i have no cows']).toarray()
Out:
array([[1, 1, 1, 0, 1, 0, 1],
       [1, 1, 0, 1, 1, 1, 0]], dtype=int64)

In : vect.vocabulary_
Out:
{'cows': 0,
 'have': 1,
 'have cows': 2,
 'have no': 3,
 'no': 4,
 'no cows': 5,
 'no have': 6}



In : from scipy.spatial.distance import euclidean

In : vect = CountVectorizer(ngram_range=(3,3), analyzer='char_wb')

In : n1, n2, n3, n4 = vect.fit_transform(['иванов', 'петров', 'петренко', 'смит']).toarray()
```

```
In : euclidean(n1, n2)
Out: 3.1622776601683795

In : euclidean(n2, n3)
Out: 2.8284271247461903

In : euclidean(n3, n4)
Out: 3.4641016151377544
```

```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from scipy.misc import face
import numpy as np

resnet_settings = {'include_top': False, 'weights': 'imagenet'}
resnet = ResNet50(**resnet_settings)

img = image.array_to_img(face())
# какой милый енот!
img = img.resize((224, 224))
# в реальной жизни может понадобиться внимательнее относиться к ресайзу
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
# нужно дополнительное измерение, т.к. модель рассчитана на работу с массивом изображений

features = resnet.predict(x)



In : import pytesseract

In : from PIL import Image

In : import requests

In : from io import BytesIO

In : img = 'http://ohscurrent.org/wp-content/uploads/2015/09/domus-01-google.jpg'
# просто случайная картинка из поиска
```

```
In : img = requests.get(img)
   ...: img = Image.open(BytesIO(img.content))
   ...: text = pytesseract.image_to_string(img)
   ...:

In : text
Out: 'Google'



# на этот раз возьмем картинку от Renthop
In : img = requests.get('https://photos.renthop.com/2/8393298_6acaf11f030217d05f3a5604b9a2f70f.jpg')
   ...: img = Image.open(BytesIO(img.content))
   ...: pytesseract.image_to_string(img)
   ...:

Out: 'Cunveztible to 4}»'
```

Геоданные

```
In : import reverse_geocoder as revgc

In : revgc.search((df.latitude, df.longitude))
Loading formatted geocoded file...
Out:
[OrderedDict([('lat', '40.74482'),
              ('lon', '-73.94875'),
              ('name', 'Long Island City'),
              ('admin1', 'New York'),
              ('admin2', 'Queens County'),
              ('cc', 'US')])]
```

Дата и время

```
df['dow'] = df['created'].apply(lambda x: x.date().weekday())
df['is_weekend'] = df['created'].apply(lambda x: 1 if x.date().weekday() in (5, 6) else 0)
```

StandartScaling

```
In : from sklearn.preprocessing import StandardScaler

In : from scipy.stats import beta

In : from scipy.stats import shapiro
```

```
In : data = beta(1, 10).rvs(1000).reshape(-1, 1)

In : shapiro(data)
Out: (0.8783774375915527, 3.0409122263582326e-27)
# значение статистики, p-value

In : shapiro(StandardScaler().fit_transform(data))
Out: (0.8783774375915527, 3.0409122263582326e-27)
# с таким p-value придется отклонять нулевую гипотезу о нормальности данных
```