Школа машинного обучения

Физтех-Школа Прикладной математики и информатики МФТИ

Лаборатория нейронных сетей и глубокого обучения (DeepHackLab)

Дедлайн: 4 апреля 23: 59 (MSK)

Домашнее задание 2

Метод k-ближайших соседей

In [1]:
```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.base import BaseEstimator
import matplotlib.pyplot as plt
import collections
import numpy as np

import scipy.stats as sps
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from matplotlib.colors import ListedColormap
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
```

Вопрос: Почему важно, чтобы в тестовой и обучающей выборке пропорции классов были максимально похожи? В качестве примера рассмотрите Ирисы Фишера: в нём три класса, каждый занимает по трети датасета. Далее мы разделяем выборку в пропорциях 2:1, например, при кросс-валидации с тремя фолдами. Что может пойти не так?

В этом ноутбуке за равенство классов отвечает StratifiedKFold (см. пример ниже, подробнее: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html))

Ответ: Нужно чтобы плотность класса в выборках была примерно равной, иначе какой-то класс можно просто не разметить

Задание 1 Примените kNN к классическому набору данных "Ирисы Фишера" (https://ru.wikipedia.org/wiki/%D0%98%D1%80%D0%B8%D1%81%D1%8B_%D0%A4%D0
Подберите оптимальное число k с помощью поиска по сетке и кросс-валидации. Постройте график зависимости качества от k. Используйте метрику accuracy.

In [2]:
```
data = load_iris()

X = data.data
y = data.target
```

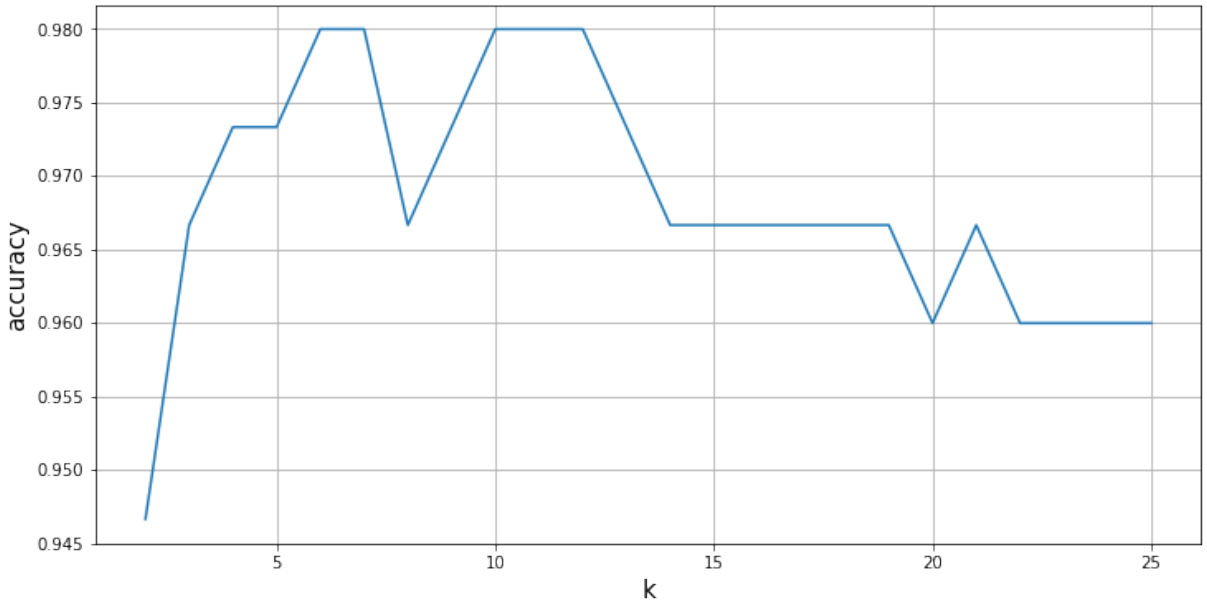
In [3]:
```
X[:5]
```

Out[3]:
```
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2]])
```

In [4]:
```
# Вы уже умеете пользоваться GridSearchCV (см семинар по knn)

model = KNeighborsClassifier()
params = {'n_neighbors': np.array(np.linspace(2, 25, 23), dtype='int')}
gscv = GridSearchCV(model, params, cv=StratifiedKFold(shuffle = False,
gscv.fit(X,y)
```

Out[4]:
```
GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=None, shuff
le=False),
       error_score='raise',
       estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30
, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform'),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'n_neighbors': array([ 2,  3,  4,  5,  6,  7,  8,
9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
       19, 20, 21, 22, 23, 25])},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn
',
       scoring='accuracy', verbose=0)
```

In [5]:
```
plt.figure(figsize=(12, 6))
plt.plot(params['n_neighbors'], gscv.cv_results_["mean_test_score"])
plt.xlabel("k", fontsize=15)
plt.ylabel("accuracy", fontsize=15)
plt.grid()
plt.show()
```



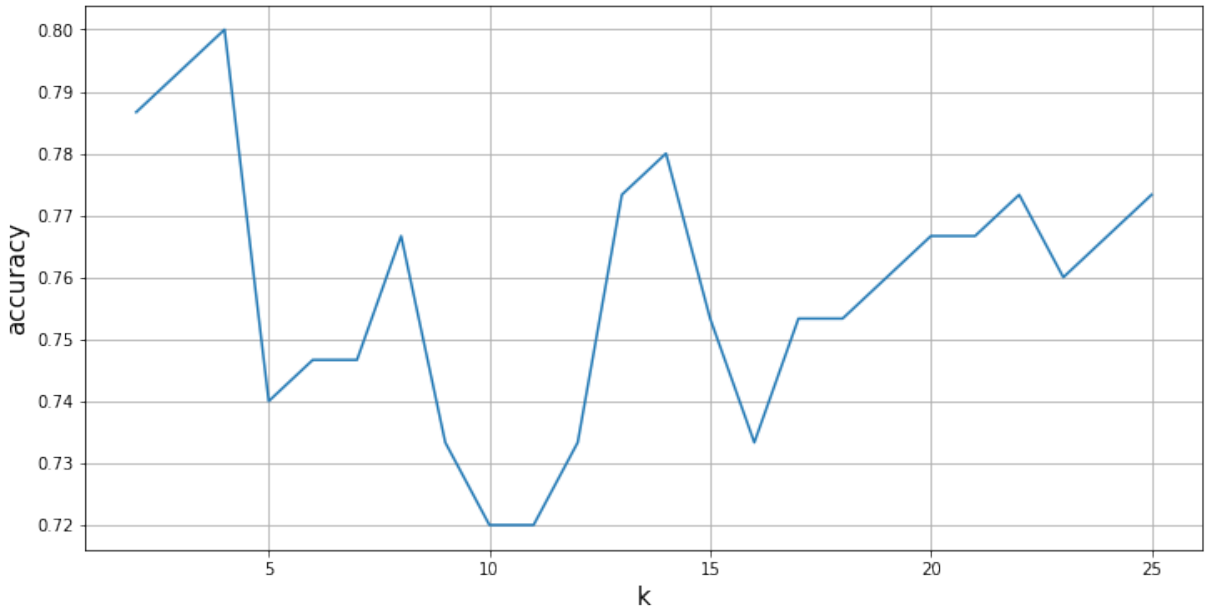
UPD: Допилил чтобы было видно для каких k

Если вы смотрели в данные, то вы видели, что признаки примерно одного порядка, т.к. это длины и ширины в сантиметрах. Предположим теперь, что один из признаков измерялся в десятых долях миллиметра. Точно так же подберите оптимальное k для новых данных, сравните качество и постройте график. Что нужно делать, чтобы такая проблема не возникала?

In [6]:
```
X_new = X * np.array((100, 1, 1, 1))
X_new[:5]
```

Out[6]:
```
array([[  5.10000000e+02,   3.50000000e+00,   1.40000000e+00,
          2.00000000e-01],
       [  4.90000000e+02,   3.00000000e+00,   1.40000000e+00,
          2.00000000e-01],
       [  4.70000000e+02,   3.20000000e+00,   1.30000000e+00,
          2.00000000e-01],
       [  4.60000000e+02,   3.10000000e+00,   1.50000000e+00,
          2.00000000e-01],
       [  5.00000000e+02,   3.60000000e+00,   1.40000000e+00,
          2.00000000e-01]])
```

In [7]:
```
# Вы уже умеете пользоваться GridSearchCV (см семинар по knn)

model = KNeighborsClassifier()
params = {'n_neighbors': np.array(np.linspace(2, 25, 23), dtype='int')}
gscv = GridSearchCV(model, params, cv=StratifiedKFold(shuffle = False,
gscv.fit(X_new,y)
plt.figure(figsize=(12, 6))
plt.plot(params['n_neighbors'], gscv.cv_results_["mean_test_score"])
plt.xlabel("k", fontsize=15)
plt.ylabel("accuracy", fontsize=15)
plt.grid()
plt.show()
```



Задание 2: Реализуйте kNN. Сравните скорости работы реализации с distance_slow, distance_fast с реализацией из sklearn. Проверьте, что качество такое же. Считать, что интерфейс fit и predict такой же, как у KNeighborsClassifier из sklearn

In [8]:
```
def distance_slow(v, a_list):
    """
    Функция, по вектору v и списку векторов a
    находящая попарные расстояния v <-> a[i]
    и возвращающая их как numpy.ndarray той же длины,
    что и список a
    (Работает медленно)
    """
    result = []
    for i in range(a_list.shape[0]):
        length = 0.
        for j in range(a_list.shape[1]):
            length += (v[j] - a_list[i, j]) ** 2
        result.append(length)
    return np.array(result)
```

In [9]:
```
def distance_fast(v, a_list):
    """
    Аналог distance_slow. Использует numpy, работает быстро.
    """
    return np.sqrt(np.sum((a_list-v)**2, axis=1))
```

In [10]:
```
class kNNClassifier:
    def __init__(self, k=3, distance = distance_slow):
        """
        Parameters
        ----------
        k: int
            Число соседей

        distance: *alias
            функция, по вектору v и списку векторов a
            находящая попарные расстояния v <-> a[i]
            и возвращающая их как numpy.ndarray той же длины,
            что и список a
        """
        self._k = k
        self._distance = distance

    def fit(self, X_train, y_train):
        self._X = np.copy(X_train)  # Копируем данных, чтобы они не пер
        self._y = np.copy(y_train)
        return self

    def predict(self, X_test):
        X_test = np.array(X_test)
        predictions = []
```

```
        objects_count = len(X_test)
        for i in range(objects_count):
            pairwise_distances = self._distance(X_test[i], self._X)

            k_nearest = self._y[np.argsort(pairwise_distances)[:self._k

            unique_values, counts = np.unique(k_nearest, return_counts=

            # Если вы что-то не понимаете - у numpy замечательная докум

            prediction = unique_values[np.argmax(counts)]
            # Предсказываем класс, представителей которого больше всего

            predictions.append(prediction)

        return predictions

    def get_params(self, deep=False):
        """
        Функция, необходимая для работы GridSearchCV
        Возвращает параметры данного экземпляра класса
        """
        return {
            "k": self._k,
            "distance": self._distance
        }

    def set_params(self, **params):
        """
        Функция, необходимая для работы GridSearchCV
        Устанавливает параметры из params
        (В данном случае пересоздаёт экземпляр класса
        и возвращает его)
        """
        self.__init__(**params)
        return self
```

In [11]:
```
clf = kNNClassifier(k = 3)
cross_val_score(clf, X, y, cv=StratifiedKFold(shuffle = False), scoring
```

Out[11]:
```
array([ 0.98039216,  0.96078431,  1.        ])
```

In [12]:
```
%%time
clf = kNNClassifier()
params = {
    "k":[1, 3, 5, 7, 9, 11, 13, 15, 17],
    "distance":[distance_slow]
}
gscv = GridSearchCV(clf, params, cv=StratifiedKFold(shuffle = False, n_
gscv.fit(X, y)
print("Best params: {}. Best score: {}".format(gscv.best_params_, gscv.
```

```
Best params: {'distance': <function distance_slow at 0x1a1064fd90>,
'k': 7}. Best score: 0.98
CPU times: user 3.17 s, sys: 20.1 ms, total: 3.19 s
Wall time: 3.21 s
```

In [13]:
```
%%time
clf = kNNClassifier()
params = {
    "k":[1, 3, 5, 7, 9, 11, 13, 15, 17],
    "distance":[distance_fast]
}
gscv = GridSearchCV(clf, params, cv=StratifiedKFold(shuffle = False, n_
gscv.fit(X, y)
print("Best params: {}. Best score: {}".format(gscv.best_params_, gscv.
```

```
Best params: {'distance': <function distance_fast at 0x1a1064fa60>,
'k': 7}. Best score: 0.98
CPU times: user 332 ms, sys: 4.85 ms, total: 337 ms
Wall time: 335 ms
```

In [14]:
```
%%time
clf = KNeighborsClassifier(n_neighbors=3)
params = {
    "n_neighbors":[1, 3, 5, 7, 9, 11, 13, 15, 17],
}
gscv = GridSearchCV(clf, params, cv=StratifiedKFold(shuffle = False, n_
gscv.fit(X, y)
```

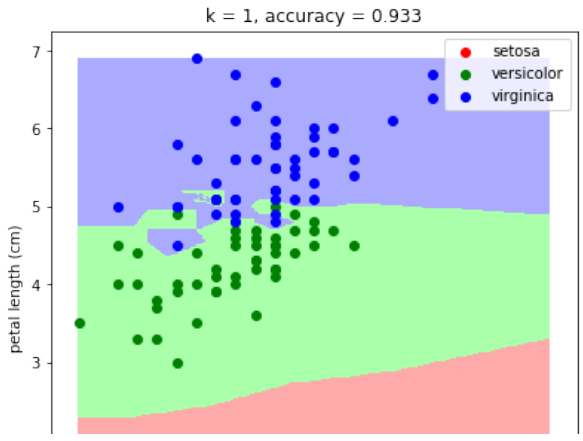
```
CPU times: user 109 ms, sys: 4.24 ms, total: 113 ms
Wall time: 112 ms
```

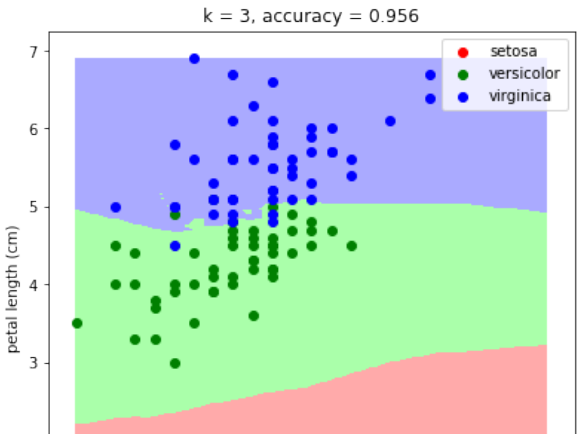
Комментарий: Интересующиеся могут изучить kd-tree (https://habrahabr.ru/post/312882/), позволяющее рассматривать меньшее число расстояний. Так же эта структура используется для отрисовки компьютерной графики.

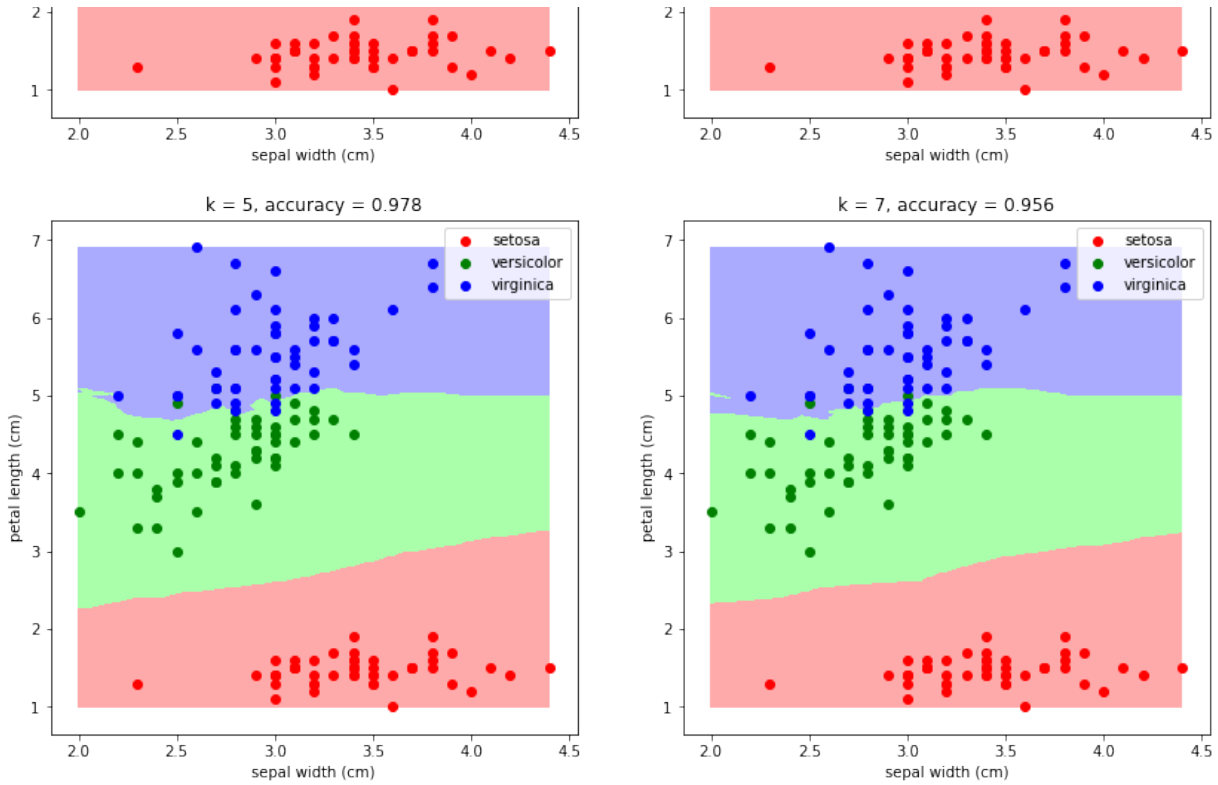
Задача 3: Пусть в данных предыдущей задачи мы получили измерения только двух признаков. Тогда признаки одного обьекта можно представить как точку на плоскости, которой в соответствие поставлен некоторый класс (можно визуализировать это как цвет). Постройте графики, изображающие принадлежность всех точек плоскости к классам для различных k.

In [15]:
```
data = load_iris()
X = data.data[:, [False, True, True, False]]
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [16]:
```
cmap = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
plt.figure(figsize=(14, 14))
for i, k in enumerate([1, 3, 5, 7]):
    plt.subplot(221 + i)

    h = 400
    xx, yy = np.meshgrid(
        np.linspace(X[:, 0].min(), X[:, 0].max(), h),
        np.linspace(X[:, 1].min(), X[:, 1].max(), h)
    )

    X_grid = np.c_[xx.ravel(), yy.ravel()]
    # Обучите (k)NN на данных X_train, y_train
    clf = kNNClassifier(k, distance_fast)
    clf.fit(X_train, y_train)
    Z = np.array(clf.predict(X_grid))
    acc = accuracy_score(clf.predict(X_test), y_test)
    # http://scikit-learn.org/stable/modules/generated/sklearn.metrics

    plt.title("k = {}, accuracy = {}".format(k, round(acc, 3)))
    zz = np.array(Z).reshape(xx.shape)

    # Вызовите plt.pcolormesh для точек xx, yy, zz и цветовой схемы cm
    # https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.pcolorme
    plt.pcolormesh(xx, yy, zz, cmap=cmap)


    plt.xlabel(data["feature_names"][1])
    plt.ylabel(data["feature_names"][2])
    for i in range(3):
        plt.scatter(X[y == i, 0], X[y == i, 1], color=["red", "green",
    plt.legend()

plt.show()
```







Как меняется форма разделяющей кривой при увеличении k и как это сказывается на тестовом качестве? Не забывайте, что иногда в данных встречаются недостоверные измерения, вызванные множеством факторов, например, проблемами при переводе данных из одного формата в другой, в том числе при занесении непосредственных измерений в компьютер.

Задание 4*: Предлагается датасет, состоящий из писем на две тематики. Задача - научиться классифицировать письма по темам.

In [141]:
```
categories = [
    'rec.autos',  # тем больше, чем две. Попробуйте другие.)
    'sci.space',
]

dataset = fetch_20newsgroups(subset='all', categories=categories,
                             shuffle=True, random_state=42)

X = dataset.data
y = dataset.target
```

In [142]:
```
len(X), len(y), y
```

Out[142]:
```
(1977, 1977, array([1, 0, 0, ..., 0, 1, 0]))
```

Рассмотрим два письма из выборки

In [143]:
```
print(X[-2])  # Класс 1
```

```
From: prb@access.digex.net (Pat)
Subject: Re: HST Servicing Mission Scheduled for 11 Days
Organization: Express Access Online Communications USA
Lines: 14
NNTP-Posting-Host: access.digex.net

In article <C6A2At.E9z@zoo.toronto.edu> henry@zoo.toronto.edu (Henry
Spencer) writes:
>
>No, the thing is designed to be retrievable, in a pinch.  Indeed, t
his
>dictated a rather odd design for the solar arrays, since they had t
o be
>retractable as well as extendable, and may thus have indirectly con
tributed
>to the array-flapping problems.


Why not design the solar arrays to be detachable.  if the shuttle is
going
to retunr the HST,  what bother are some arrays.  just fit them with
a quick release.

one  space walk,  or use the second canadarm to remove the arrays.

pat
```

In [144]:
```
print(X[2])  # Класс 0
```

```
From: aas7@po.CWRU.Edu (Andrew A. Spencer)
Subject: Re: MR2 - noisy engine.
Organization: Case Western Reserve University, Cleveland, OH (USA)
Lines: 33
Reply-To: aas7@po.CWRU.Edu (Andrew A. Spencer)
NNTP-Posting-Host: slc5.ins.cwru.edu


In a previous article, eliot@lanmola.engr.washington.edu (eliot) say
s:

>In article <1r1vofINN871@usenet.pa.dec.com> tomacj@opco.enet.dec.co
m (THUNDERBIRDS ARE GO !!!) writes:
>>      Are there any MR2 owners or motor-head gurus out there, that
know why
>>my MR2's engine sounds noisy? The MR2's engine is noisy at the bes
t of times,
>>but not even a nice nose - it's one of those very ugly noises.
>
>assuming yours is a non turbo MR2, the gruffness is characteristic
of
```

```
>a large inline 4 that doesn't have balance shafts.  I guess toyota
>didn't care about "little" details like that when they can brag abo
ut
>the mid engine configuration and the flashy styling.
>
>myself, i automatically cross out any car from consideration (or
>recommendation) which has an inline 4 larger than 2 liters and no
>balance shafts..  it is a good rule of thumb to keep in mind if you
>ever want a halfway decent engine.
>
>if the noise really bugs you, there is nothing else that you can do
>except to sell it and get a V6.
>
>
>eliot

nice theory.  too bad the MR2's never came with a four cylinder over
2.0
liters.  More like 1.6.  Or did they? were the nonturbo MR2II's  2.2
or
some such?

I also understand that anyone using balancing shafts on four cylinde
rs, must
pay SAAB a royalty for using their patented design..like Porsche's 3
.0 I4...

c ya
DREW
```

В целом, после прочтения понятно, что первое письмо про космос а второе - про машины. Для того, чтобы классифицировать тексты, нужно перевести их в удобный для алгоритма вид, т.е. сделать из письма вектор. Прделагается делать это так: составить список всех используемых слов. Зафиксировать число N самых популярных слов, которые мы будет рассматривать. Каждому письму сопоставлять вектор длины N следующего вида: в a[i] записано число вхождений i-го по популярности слова. Данную задачу решает CountVectoizer: используя его преобразуйте тексты в векторы и подсчитайте качество (accuracy) на классификации.

In [117]:
```
vect = CountVectorizer(max_features=50)
vect.fit(X)
Xt = vect.transform(X)
```

In [ ]:
```
# Проверьте качество kNN на данных Xt, y.  Метрика - accuracy
```

Очевидна проблема: самые часто встречающиеся слова встречаются одинаково часто во всех текстах: это a, the, и прочие. Для этой проблемы также существует стандартное решение: Проверьте качество теперь:

In [ ]:
```
# Создайте экземпляр CountVectorizer с параметрами max_features=50, st
# Как в примере выше, преобразуйте X в Xt
```

In [ ]:
```
# Проверьте качество kNN на данных Xt, y.  Метрика - accuracy
```

Нетрудно догадаться, какие слова будут самыми популярными:

In [124]:
```
vect.transform(["cake space space car car car"]).toarray()
```

Out[124]:
```
array([[0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
0, 0,
        0, 0, 0, 0, 0, 0]])
```

Качество метрических алгоритмов очень сильно зависит от метрики (функции расстояния). Сравните качество метрик "minkowski" и "cosine" - последняя считает величину, численно равную единице минут косинус угла между векторами, что более уместно для текстов, что обычная евклидова метрика (metric="minkowski", p=2). Найдите параметры, на которых достигается лучшее качество.

In [ ]:
```
params = {
    "n_neighbors": range(1, 15, 2),
    "metric": ["minkowski", "cosine"]
}

# Вы знаете, что делать.)
```

In [ ]:
```

```

Замечание: Известна проблема несовместимости версий sklearn: метрика "cosine" не доступна в старых версиях sklearn.

Учитывая, что самый важный навык в IT-профессиях это умение пользоваться поисковиком (и читать технические тексты, например, документации), предлагаются следующие варианты:

- использовать sklearn версии 0.19.1 (вы быстро разберётесь, как узнать текущую версию и как её обновить, это полезно знать)
- написать метрику cosine_distances для своего класса и использовать её.

  В sklearn реализована похожая функция, можете использовать её, но никто не запрещает написать свою реализацию. Обратите внимание, что CountVectorizer возвращает объект класса scipy.sparse.csr.csr_matrix, к чему вы, наверное, не были готовы при реализации класса, так что нужно преобразовать Xt к numpy.ndarray (документация в помощь)

  Обьект класса scipy.sparse.csr.csr_matrix используется так как матрицы, возвращаемые CountVectorizer, обычно почти полностью состоят из нулей и использование разреженных матриц сильно экономит оперативную память. Такие матрицы имеют схожие с numpy.ndarray интерфейсы, но совместимость не полная.

При большом числе признаков метрические алгоритмы обычно плохо работают, подробнее: проклятие размерности (https://en.wikipedia.org/wiki/Curse_of_dimensionality).