Семинар: Терминология. Введ sklearn. KNN. Задачи

Задание 1: kNN. Визуализация решающих г kNN.

В этом задании мы изобразим решающую поверхность для чтобы наглядно увидеть, как классификатор принимает ре Для простоты будем работать со встроенным в `sklearn` н содержащим информацию о характеристиках трёх видов в можно найти здесь (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_win и здесь (https://rdrr.io/cran/rattle.data/man/wine.html).

Загрузим набор данных и сохраним информацию о призна зависимой переменной – в переменную `y`.

В [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

В [2]:
```
from sklearn.datasets import load_wine

data = load_wine()
X = pd.DataFrame(data['data'], columns = data[
y = data['target']
X.head(8)
```

Out[2]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	to
0	14.23	1.71	2.43	15.6	127.0	
1	13.20	1.78	2.14	11.2	100.0	
2	13.16	2.36	2.67	18.6	101.0	
3	14.37	1.95	2.50	16.8	113.0	
4	13.24	2.59	2.87	21.0	118.0	
5	14.20	1.76	2.45	15.2	112.0	
6	14.39	1.87	2.45	14.6	96.0	
7	14.06	2.15	2.61	17.6	121.0	

В [3]:
```
X.shape
```

Out[3]:
```
(178, 13)
```

Задача 1.1. Есть ли в наборе данных пропущенные значен
Есть ли в наборе данных категориальные переменные? Ес
при помощи OneHot-кодирования.

В [4]:
```
X.isna().mean() # Тут мы измеряем соотношение
                # что пропущенных значений у н
```

Out[4]:
```
alcohol                         0.0
malic_acid                      0.0
ash                             0.0
alcalinity_of_ash               0.0
magnesium                       0.0
total_phenols                   0.0
flavanoids                      0.0
nonflavanoid_phenols            0.0
proanthocyanins                 0.0
color_intensity                 0.0
hue                             0.0
od280/od315_of_diluted_wines    0.0
proline                         0.0
dtype: float64
```

В [5]:
```
X.dtypes == "object" # Тут мы проверяем, есть
                     # (типа object), значит н
                     # значит у нас нет катего
```

Out[5]:
```
alcohol                         False
malic_acid                      False
ash                             False
alcalinity_of_ash               False
magnesium                       False
total_phenols                   False
flavanoids                      False
nonflavanoid_phenols            False
proanthocyanins                 False
color_intensity                 False
hue                             False
od280/od315_of_diluted_wines    False
proline                         False
dtype: bool
```

В [6]:
```
# Нам нечего кодировать (((
```

Задача 1.2. Используя функцию `train_test_split()` , ра
тренировочную и тестовую, и долю тестовой выборки зада
разбиение осуществляется случайным образом, не забуды
`np.random.seed()` для воспроизводимости результатов.

В [7]:
```
from sklearn.model_selection import train_test
```

В [8]:
```
X_train, X_test, y_train, y_test = train_test_
# разбиваем фреймы X и y на test и train с дол
```

В [9]:
```
X_train.shape    # проверяем размер X_train. Д
```

Out[9]:
```
(124, 13)
```

В [10]:
```
X_test.shape    # Проверяем длину X_test. Она
```

Out[10]:
```
(54, 13)
```

Задача 1.3. На тренировочной выборке обучите шесть кла
отличающихся только числом соседей. Для первого класси
поставьте равным 1, для второго - 3, для третьего – 5, для
– 15 и для шестого – 25 (обратите внимание на параметр
`KNeighborsClassifier`). Для обучения используйте толы
и `magnesium` – и евклидово расстояние. Не забудьте масш
например, при помощи модуля `StandardScaler`.

Выведите долю правильных ответов на тренировочной и те
каждого классификатора.

В [11]:
```
from sklearn.preprocessing import StandardScal
from sklearn.neighbors import KNeighborsClassi
```

В [12]:
```
X_train_1_3 = X_train[['alcohol', 'magnesium']
X_test_1_3 = X_test[['alcohol', 'magnesium']]
```

В [29]:
```
scaler = StandardScaler()
scaler.fit(X_train_1_3)
scaled_X_train_1_3 = scaler.transform(X_train_
scaled_X_test_1_3 = scaler.transform(X_test_1_
scaled_X_train_1_3[:15]   # вывожу первые 15,
```

Out[29]:
```
array([[-1.10453814, -1.29401378],
       [-0.60884898,  4.7936089 ],
       [ 1.17054803,  0.66003794],
       [-1.37144769, -1.21885795],
       [-0.44361925, -0.31698792],
       [ 0.20458966, -1.51948129],
       [ 0.12832978, -0.9182346 ],
       [ 1.11970812,  1.26128463],
       [ 1.05615822, -0.61761126],
       [-1.10453814, -2.12072797],
       [-0.9520184 , -0.16667625],
       [ 2.28902615, -0.54245543],
       [-0.8884685 , -1.36916962],
       [ 1.6026873 ,  0.96066128],
       [-0.83762859, -0.76792293]])
```

B [18]:

```
num_neighbours = [1, 3, 5, 10, 15, 25]

tab = pd.DataFrame(columns=['NN', 'Train', 'Te

classifiers = []

for i in num_neighbours:
    clf = KNeighborsClassifier(n_neighbors=i,


    clf.fit(scaled_X_train_1_3, y_train)

    y_train_predicted = clf.predict(scaled_X_t
    y_test_predicted = clf.predict(scaled_X_te

    raw = pd.Series({'NN': i, 'Train': np.mean
    # здесь мы считаем точность предсказаний и
    tab = pd.concat([tab, raw.to_frame().T], i
    classifiers.append(clf)
tab
```

Out[18]:

	NN	Train	Test
0	1.0	0.991935	0.685185
1	3.0	0.814516	0.685185
2	5.0	0.798387	0.722222
3	10.0	0.741935	0.777778
4	15.0	0.677419	0.814815
5	25.0	0.709677	0.796296

B [24]:

```
################################### Пример (не
```

Задача 1.4. Установите библиотеку `mlxtend` командой ни
можно установить из терминала при помощи `pip` или co
(http://rasbt.github.io/mlxtend/installation/).

In [15]:
```
!pip install mlxtend
```

```
Requirement already satisfied: mlxtend in c:\users
e-packages (0.23.0)
Requirement already satisfied: scipy>=1.2.1 in c:\
b\site-packages (from mlxtend) (1.10.1)
Requirement already satisfied: numpy>=1.16.2 in c:
ib\site-packages (from mlxtend) (1.24.3)
Requirement already satisfied: pandas>=0.24.2 in c
\lib\site-packages (from mlxtend) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.2
nda3\lib\site-packages (from mlxtend) (1.3.0)
Requirement already satisfied: matplotlib>=3.0.0 i
a3\lib\site-packages (from mlxtend) (3.7.1)
Requirement already satisfied: joblib>=0.13.2 in c
\lib\site-packages (from mlxtend) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in
3\lib\site-packages (from matplotlib>=3.0.0->mlxte
Requirement already satisfied: cycler>=0.10 in c:\
b\site-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: fonttools>=4.22.0 i
a3\lib\site-packages (from matplotlib>=3.0.0->mlxt
Requirement already satisfied: kiwisolver>=1.0.1 i
a3\lib\site-packages (from matplotlib>=3.0.0->mlxt
Requirement already satisfied: packaging>=20.0 in
\lib\site-packages (from matplotlib>=3.0.0->mlxten
Requirement already satisfied: pillow>=6.2.0 in c:
ib\site-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: pyparsing>=2.3.1 in
3\lib\site-packages (from matplotlib>=3.0.0->mlxte
Requirement already satisfied: python-dateutil>=2.
onda3\lib\site-packages (from matplotlib>=3.0.0->m
Requirement already satisfied: pytz>=2020.1 in c:\
b\site-packages (from pandas>=0.24.2->mlxtend) (20
Requirement already satisfied: threadpoolctl>=2.0.
onda3\lib\site-packages (from scikit-learn>=1.0.2-
Requirement already satisfied: six>=1.5 in c:\user
te-packages (from python-dateutil>=2.7->matplotlib
0)
```

Если всё прошло успешно, то в выводе команды выше вы
"successfully installed", а следующая ячейка выполнится бе

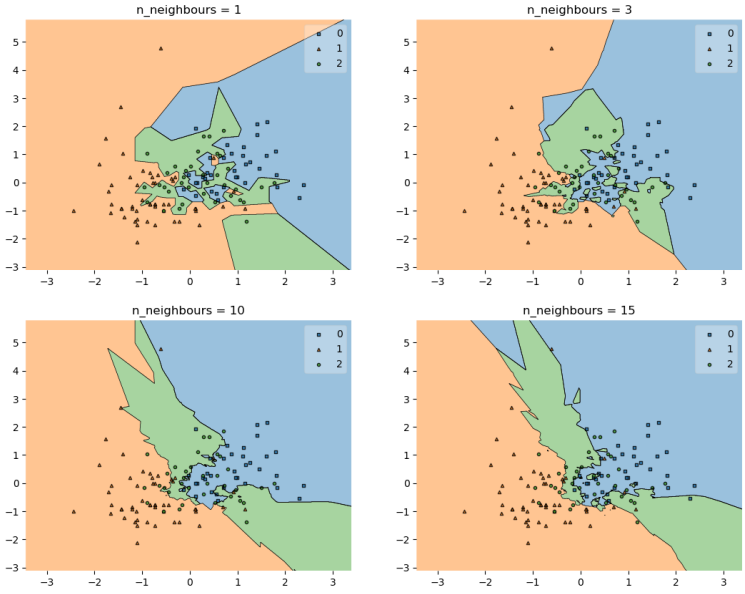
In [20]:
```
from mlxtend.plotting import plot_decision_reg
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import itertools
```

Задача 1.5. Библиотека `mlxtend` позволяет достаточно пр
решающие поверхности обученных классификаторов. Изуч
(http://rasbt.github.io/mlxtend/user_guide/plotting/plot_decisior
найдите, как можно построить несколько графиков решающ
(decision region grids). Постройте такую сетку графиков для
классификаторов.

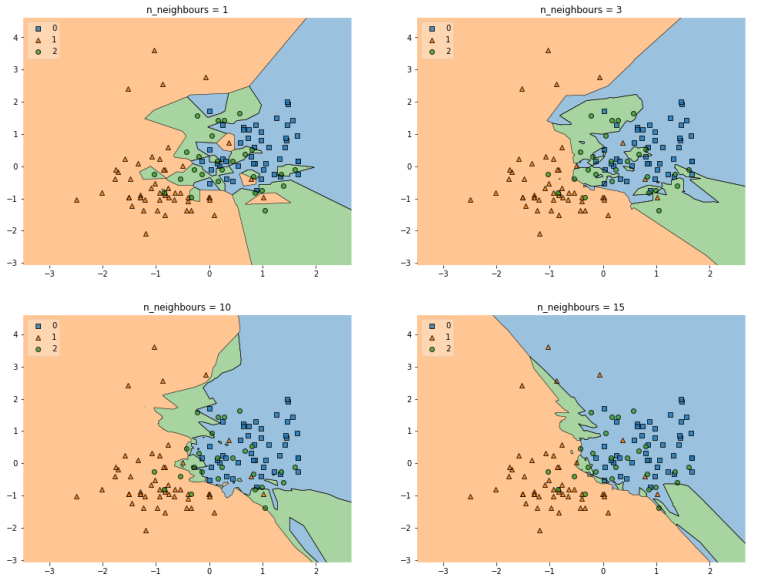
Подсказки:

1. Вы можете использовать готовый код, приведённый в д
адаптировать его для нашего случая.
2. Вам могут понадобиться дополнительные библиотеки,
примере из документации.
3. Обратите внимание на то, как нужно изменить параме
и `itertools.product()` для нашего числа классифик
4. В функции `plot_decision_region()` используйте y_
из `X_train` . Возможно, их придётся перевести в форм
5. Если в задаче 1.3 вы сохраните обученные классифик
необходимости обучать их заново.
6. Построение графика может занять некоторое время –
подождать!

In [24]:
```
gs = gridspec.GridSpec(ncols=3, nrows=2)  # co
fig = plt.figure(figsize=(20,10))         # on
font = {'size': 10}                       # ук
lines = {'markersize': 3}                 # ук

plt.rc('font', **font)                    # до
plt.rc('lines', **lines)

labels = ['n_neighbours = {}'.format(i) for i

for clf, lab, grd in zip(classifiers, labels,
    # зип объединяет объекты с соответствующим

    ax = plt.subplot(gs[grd[0], grd[1]]) # в i
                                         # под
                                         # из

    fig = plot_decision_regions(X = scaled_X_t
    # в X передаем данные (см. задание 1.3), к
    # метки и в clf наши классификаторы.

    plt.title(lab)  # даем заголовок

plt.show()          # выводим
```



In [ ]:
```
1 ################################## Пример (не
```



Задача 1.6 Прокомментируйте результаты, полученные в з
число соседей оптимально использовать для обучения кла
ваш выбор при помощи описания геометрии данных и полу
поверхности.

Type *Markdown* and LaTeX: α^2

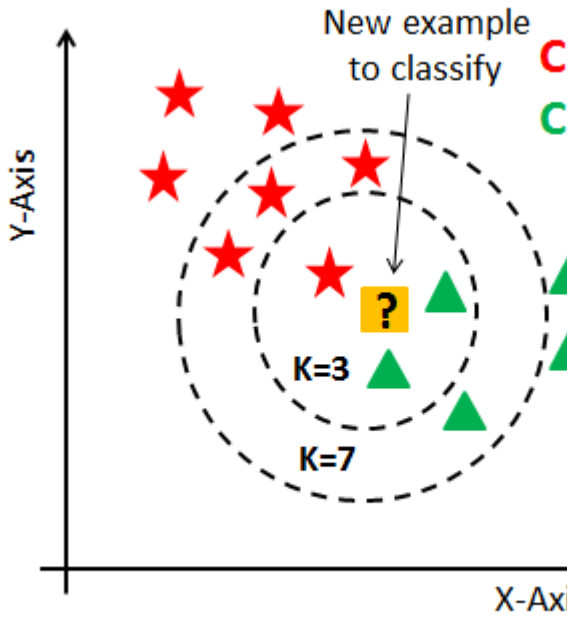
В задании 1.3 мы получили таблицу, отражающую долю пр
test выборках, в зависимости от изменения гиперпараметр
количество соседей. Видно, что на train выборке точность
что не ровно 1), а в дальше она начинает падать, потому ч
своей группы объекты(шумовые) попадают уже в другие гр
ситуация противоположная. С увеличение числа соседей,
определенного момента), потому что модель становится на
Падать точность начинает, когда число соседей становится
то класс забирает себе чужие области, потому что его элем
train выборке. Например, если взять число соседей равное
выборке, то все последующие элементы будут относится т
элементов которого больше.

В задании 1.5 видно, что разделяющая поверхность с увел
становится проще, так как на нее меньше действуют шумо
взгляд, самыми лучшими тут будут поверхности с 10 и 15 с

Для того, чтобы оценить качество модели, нам нужно посм
тестовой выборке, потому что именно они показывают, как
на данных, которые она еще не видела. Исходя из результа
в задании 1.3, наиболее подходящее количество соседей д
15. Этот ответ подтверждает и то, что разделяющая повер
выглядит разумно.

Задание 2. KNN своими руками (дополните

В данном задании мы попробуем реализовать алгоритм KN
данном случае мы попробуем сделать KNN для классифик



В [40]:
```
1 import numpy as np
2 from collections import Counter  #не пригодило
```

B [25]:

```
import numpy as np
from collections import Counter
import statistics as st

class KNN:
    def __init__(self, k:int):  # конструктор
        self.k = k

    def fit(self, X, Y):        # функция обуч
        self.X_train = X        # Значит нам н
        self.y_train = Y
        self.m, self.n = self.X_train.shape  #

    def predict(self, X_test):  # метод предск
        self.X_test = X_test    # сохраняем те
        self.m_test, self.n = self.X_test.shap


        y_predict = np.zeros(self.m_test)

        for i in range(self.m_test):
            distances = np.zeros(self.m)

            for j in range(self.m):        # б
                distances[j]=np.sqrt((np.sum(n

            neighbors = self.y_train[distances

            y_predict[i]= st.mode(neighbors)

        return y_predict
```

В [26]:
```python
# Не меняйте файл!
def test_knn(KNN):
    knn = KNN(k=1)
    X_train =  np.array([[1, 1], [2, 2]])
    y_train =  np.array([0, 1])
    X_test =  np.array([[1.5, 1.5]])
    knn.fit(X_train, y_train)
    assert knn.predict(X_test) == [0]

    knn = KNN(k=3)
    X_train = np.array([[1, 1], [2, 2], [3, 3]
    y_train = np.array([0, 0, 0, 0, 1, 1, 1, 1
    X_test = np.array([[9.5, 9.5]])
    knn.fit(X_train, y_train)
    assert knn.predict(X_test) == [1]

    knn = KNN(k=3)
    X_train = np.array([[1, 1], [2, 2], [3, 3]
    y_train = np.array([0, 0, 0, 0, 1, 1, 1, 1
    X_test = np.array([[5.5, 5.5]])
    knn.fit(X_train, y_train)
    assert knn.predict(X_test) == [1]

    knn = KNN(k=3)
    X_train = np.array([[1, 1], [2, 2], [3, 3]
    y_train = np.array([0, 0, 0, 0, 1, 1, 1, 1
    X_test = np.array([[15, 15]])
    knn.fit(X_train, y_train)
    assert knn.predict(X_test) == [1]

    knn = KNN(k=3)
    X_train = np.array([[1, 1], [2, 2], [3, 3]
    y_train = np.array([0, 0, 0, 0, 1, 1, 1, 1
    X_test = np.array([[5, 5], [2, 2]])
    knn.fit(X_train, y_train)
    assert all(knn.predict(X_test) == [1, 0])
```

В [27]:
```python
# Если тесты эти пройдены, то все верно!
test_knn(KNN)
```

В [28]:
```python
# Все тесты пройдены.
```