In [9]:

```
#Подключаем библиотеки
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import sklearn as sk
import numpy as np

#Подключаем искомый набор данных
from sklearn.datasets import load_digits

#Подключаем алгоритмы
from sklearn.svm import SVC                          #Опорные вектора
from sklearn.neighbors import KNeighborsClassifier #К-ближайших соседей
from sklearn.tree import DecisionTreeClassifier    #Деревья решений
from sklearn.naive_bayes import GaussianNB         #Баес
from sklearn.linear_model import LogisticRegression#Логистическая регрессия

from sklearn import metrics

#Подключаем функцию для перемешивания и разбиения данных
from sklearn.model_selection import train_test_split
digits = load_digits()
#Проводим перемешивание данных и разделение на тестовую и тренировочную выборки
X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target,
                                                    stratify = digits.target, random_st
ate = 66)

#Подключаем корректность отображения русского языка на графиках
%matplotlib inline

#Метод опорных векторов
neighbors_settings = range(1, 11)
train_acc_ov = [] #Формируем пустой массив для тренировочной выборки
test_acc_ov = []  #Формироем пустой массив для тестовой выборки
ov = SVC().fit(X_train,y_train) #Строим модель
train_acc_ov.append(ov.score(X_train,y_train)) #Записываем правильность на обучающем на
боре
test_acc_ov.append(ov.score(X_test, y_test))   #Записываем правильность на тестовом наб
оре
exov = y_train
prov = ov.predict(X_train)
print(metrics.classification_report(exov,prov))

#Метод К-ближайших соседей
train_acc_kn = []
test_acc_kn = []
kn = KNeighborsClassifier().fit(X_train,y_train)
train_acc_kn.append(kn.score(X_train,y_train))
test_acc_kn.append(kn.score(X_test, y_test))
exkn = y_train
prkn = kn.predict(X_train)
print(metrics.classification_report(exkn,prkn))

#Метод Баеса
train_acc_gn = []
test_acc_gn = []
```

```
gn = GaussianNB().fit(X_train,y_train)
train_acc_gn.append(gn.score(X_train,y_train))
test_acc_gn.append(gn.score(X_test, y_test))
exgn = y_train
prgn = gn.predict(X_train)
print (metrics.classification_report(exgn, prgn))

#Деревья решений
train_acc_dt = []
test_acc_dt = []
dt = DecisionTreeClassifier().fit(X_train,y_train)
train_acc_dt.append(dt.score(X_train,y_train))
test_acc_dt.append(dt.score(X_test, y_test))
exdt = y_train
prdt = dt.predict(X_train)
print(metrics.classification_report(exdt, prdt))

#Логистическая регрессия
train_acc_lr = []
test_acc_lr = []
lr = LogisticRegression().fit(X_train,y_train)
train_acc_lr.append(lr.score(X_train,y_train))
test_acc_lr.append(lr.score(X_test, y_test))
exlr = y_train
prlr = lr.predict(X_train)
print(metrics.classification_report(exlr, prlr))

print("Метод опорных векторов")
print("test = ", test_acc_ov)
print("train =", train_acc_ov)

print("Метод K-ближайших соседей")
print("test = ", test_acc_kn)
print("train = ", train_acc_kn)

print("Метод Баеса")
print("test = ", test_acc_gn)
print("train = ", train_acc_gn)

print("Деревья решений")
print("test = ", test_acc_dt)
print("train = ", train_acc_dt)

print("Логитическая регрессия")
print("test = ", test_acc_lr)
print("train = ", train_acc_lr)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	133
1	1.00	1.00	1.00	136
2	1.00	1.00	1.00	133
3	1.00	1.00	1.00	137
4	1.00	1.00	1.00	136
5	1.00	1.00	1.00	136
6	1.00	1.00	1.00	136
7	1.00	1.00	1.00	134
8	1.00	1.00	1.00	131
9	1.00	1.00	1.00	135
avg / total	1.00	1.00	1.00	1347

	precision	recall	f1-score	support
0	1.00	1.00	1.00	133
1	0.97	1.00	0.99	136
2	1.00	0.99	1.00	133
3	0.99	0.98	0.98	137
4	0.99	0.99	0.99	136
5	0.99	0.98	0.99	136
6	0.99	1.00	1.00	136
7	0.98	1.00	0.99	134
8	0.98	0.97	0.98	131
9	0.98	0.96	0.97	135
avg / total	0.99	0.99	0.99	1347

	precision	recall	f1-score	support
0	0.99	0.99	0.99	133
1	0.83	0.83	0.83	136
2	0.97	0.73	0.83	133
3	0.94	0.84	0.89	137
4	0.98	0.90	0.94	136
5	0.94	0.93	0.94	136
6	0.96	0.99	0.97	136
7	0.75	0.99	0.85	134
8	0.62	0.85	0.72	131
9	0.97	0.70	0.81	135
avg / total	0.90	0.88	0.88	1347

	precision	recall	f1-score	support
0	1.00	1.00	1.00	133
1	1.00	1.00	1.00	136
2	1.00	1.00	1.00	133
3	1.00	1.00	1.00	137
4	1.00	1.00	1.00	136
5	1.00	1.00	1.00	136
6	1.00	1.00	1.00	136
7	1.00	1.00	1.00	134
8	1.00	1.00	1.00	131
9	1.00	1.00	1.00	135
avg / total	1.00	1.00	1.00	1347

	precision	recall	f1-score	support

```
           0       1.00      1.00      1.00       133
           1       0.97      0.99      0.98       136
           2       1.00      1.00      1.00       133
           3       0.99      1.00      1.00       137
           4       1.00      1.00      1.00       136
           5       1.00      1.00      1.00       136
           6       1.00      1.00      1.00       136
           7       1.00      1.00      1.00       134
           8       0.97      0.96      0.97       131
           9       1.00      0.99      0.99       135

avg / total       0.99      0.99      0.99      1347

Метод опорных векторов
test =  [0.62]
train = [1.0]
Метод К-ближайших соседей
test =  [0.99555555555555553]
train =  [0.98737936154417227]
Метод Баеса
test =  [0.86222222222222222]
train =  [0.87602078693392726]
Деревья решений
test =  [0.87333333333333329]
train =  [1.0]
Логитическая регрессия
test =  [0.97777777777777775]
train =  [0.99331848552338531]
```

In [ ]:

In [ ]: