Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа №1

по дисциплине «Машинное обучение»

Выполнил
студент гр. 3530904/00103 Плетнева А.Д.

Руководитель Селин И.А.

Санкт-Петербург
2022

Оглавление

Задание 3
Задание 1 5
Задание 2 6
Задание 3 7
Задание 4 8
Пункт a 8
Пункт b 9
Пункт c 10
Пункт d 11
Пункт e 12
Задание 5 13
Пункт a 13
Пункт б 14
Задание 6 15
Текст программы 16

Задание

1. Исследуйте, как объем обучающей выборки и количество тестовых данных, влияет на точность классификации в датасетах про крестики-нолики (tic_tac_toe.txt) и о спаме e-mail сообщений (spam.csv) с помощью наивного Байесовского классификатора. Постройте графики зависимостей точности на обучающей и тестовой выборках в зависимости от их соотношения.
2. Сгенерируйте 100 точек с двумя признаками X1 и X2 в соответствии с нормальным распределением так, что одна и вторая часть точек (класс -1 и класс 1) имеют параметры: мат. ожидание X1, мат. ожидание X2, среднеквадратические отклонения для обеих переменных, соответствующие вашему варианту (указан в таблице). Построить диаграммы, иллюстрирующие данные. Построить Байесовский классификатор и оценить качество классификации с помощью различных методов (точность, матрица ошибок, ROC и PR-кривые). Является ли построенный классификатор «хорошим»?
3. Постройте классификатор на основе метода k ближайших соседей для обучающего множества Glass (glass.csv). Посмотрите заголовки признаков и классов. Перед построением классификатора необходимо также удалить первый признак Id number, который не несет никакой информационной нагрузки.
   a. Постройте графики зависимости ошибки классификации от количества ближайших соседей.
   b. Определите подходящие метрики расстояния и исследуйте, как тип метрики расстояния влияет на точность классификации.
   c. Определите, к какому типу стекла относится экземпляр с характеристиками: RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1
4. Постройте классификаторы на основе метода опорных векторов для наборов данных из файлов svmdataN.txt и svmdataNtest.txt, где N – индекс задания:
   a. Постройте алгоритм метода опорных векторов с линейным ядром. Визуализируйте разбиение пространства признаков на области с помощью полученной модели (пример визуализации). Выведите количество полученных опорных векторов, а также матрицу ошибок классификации на обучающей и тестовой выборках.
   b. Постройте алгоритм метода опорных векторов с линейным ядром. Добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения штрафного параметра. Выберите оптимальное значение данного параметра и объясните свой выбор. Всегда ли нужно добиваться минимизации ошибки на обучающей выборке?
   c. Постройте алгоритм метода опорных векторов, используя различные ядра (линейное, полиномиальное степеней 1-5, сигмоидальная функция, гауссово). Визуализируйте разбиение пространства признаков на области с помощью полученных моделей. Сделайте выводы.
   d. Постройте алгоритм метода опорных векторов, используя различные ядра (полиномиальное степеней 1-5, сигмоидальная функция, гауссово). Визуализируйте разбиение пространства признаков на области с помощью полученных моделей. Сделайте выводы.
   e. Постройте алгоритм метода опорных векторов, используя различные ядра (полиномиальное степеней 1-5, сигмоидальная функция, гауссово). Изменяя

значение параметра ядра (гамма), продемонстрируйте эффект переобучения, выполните при этом визуализацию разбиения пространства признаков на области.

5. Постройте классификаторы для различных данных на основе деревьев решений:
   a. Загрузите набор данных Glass из файла glass.csv. Постройте дерево классификации для модели, предсказывающей тип (Type) по остальным признакам. Визуализируйте результирующее дерево решения. Дайте интерпретацию полученным результатам. Является ли построенное дерево избыточным? Исследуйте зависимость точности классификации от критерия расщепления, максимальной глубины дерева и других параметров по вашему усмотрению.
   b. Загрузите набор данных spam7 из файла spam7.csv. Постройте оптимальное, по вашему мнению, дерево классификации для параметра yesno. Объясните, как был осуществлён подбор параметров. Визуализируйте результирующее дерево решения. Определите наиболее влияющие признаки. Оцените качество классификации.
6. Загрузите набор данных из файла bank_scoring_train.csv. Это набор финансовых данных, характеризующий физических лиц. Целевым столбцом является «SeriousDlqin2yrs», означающий, ухудшится ли финансовая ситуация у клиента. Постройте систему по принятию решения о выдаче или невыдаче кредита физическому лицу. Сделайте как минимум 2 варианта системы на основе различных классификаторов. Подберите подходящую метрику качества работы системы исходя из специфики задачи и определите, принятие решения какой системой сработало лучше на bank_scoring_test.csv.

Задание 1

Для выполнения данного задания воспользуемся байесовским классификатором из библиотеки sklearn.
Получаем следующую зависимость точности от доли тестовых данных:

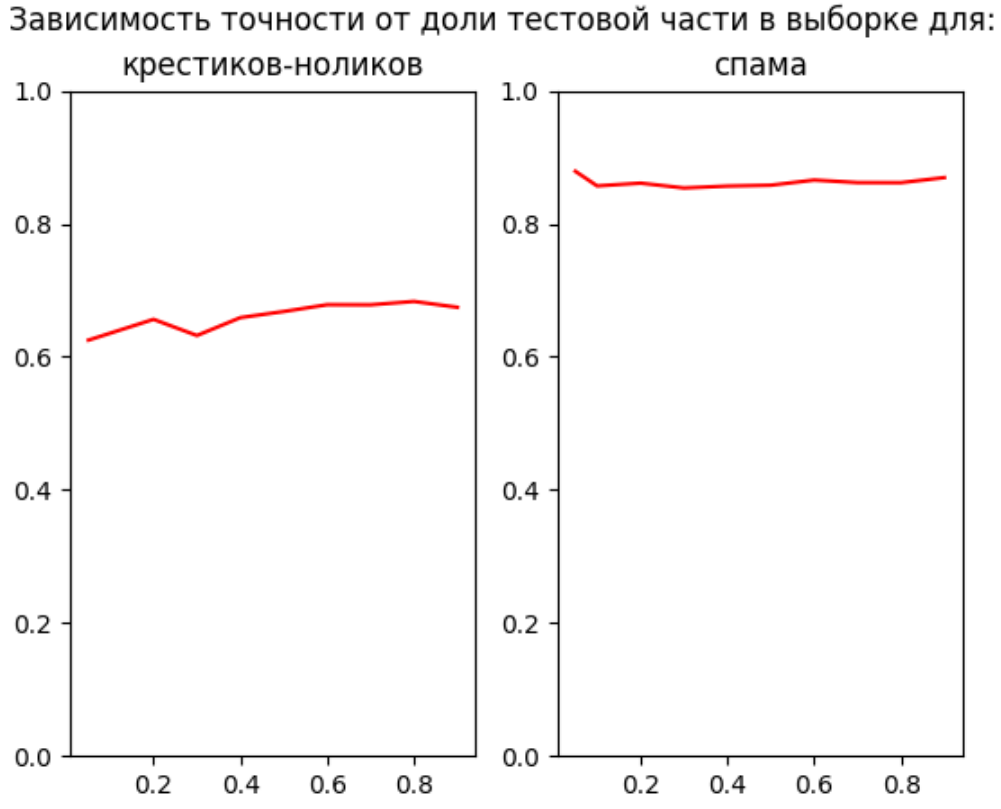
Для крестиков-ноликов

Доля тестовых данных	Точность
0.05	0.625
0.1	0.635417
0.2	0.65625
0.3	0.631944
0.4	0.658854
0.5	0.668058
0.6	0.678261
0.7	0.678092
0.8	0.683181
0.9	0.674392

Для спама

Доля тестовых данных	Точность
0.05	0.878788
0.1	0.856833
0.2	0.861021
0.3	0.853729
0.4	0.8566
0.5	0.857888
0.6	0.865628
0.7	0.861844
0.8	0.861722
0.9	0.869355

Графики:



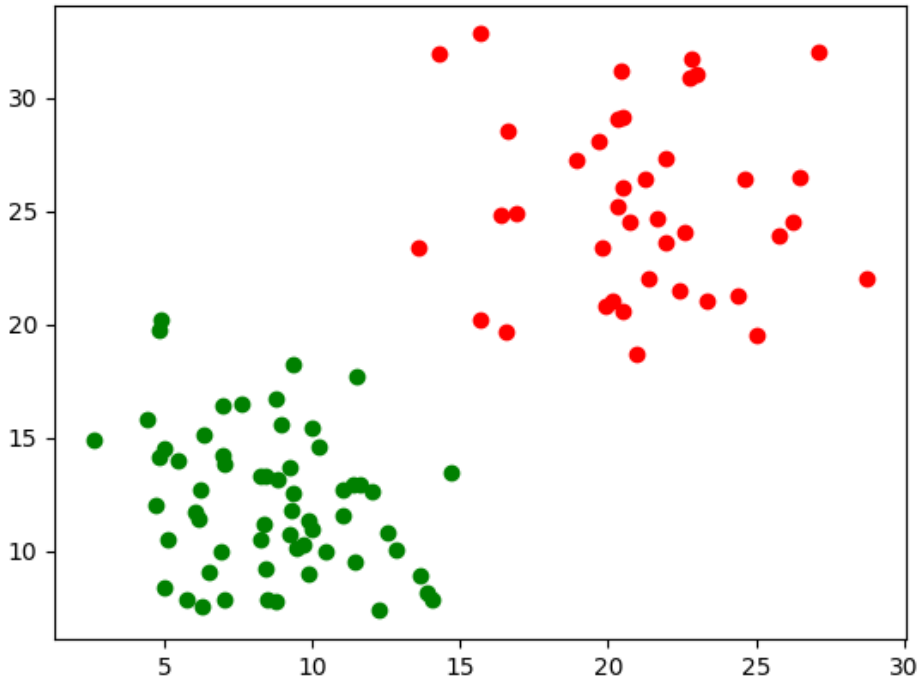
Точность крестиков-ноликов немного растет с увеличением доли тестовой выборки – то есть при малой доле тестовой выборки модель переобучается, точность для спама практически не меняется при увеличении объема тестовой выборки и соответственно уменьшении объема обучающей выборки.

Задание 2

Сгенеровали 100 точек с двумя признаками X1 и X2 в соответствии с нормальным распределением так, что одна и вторая часть точек (класс -1 и класс 1) имеют параметры:

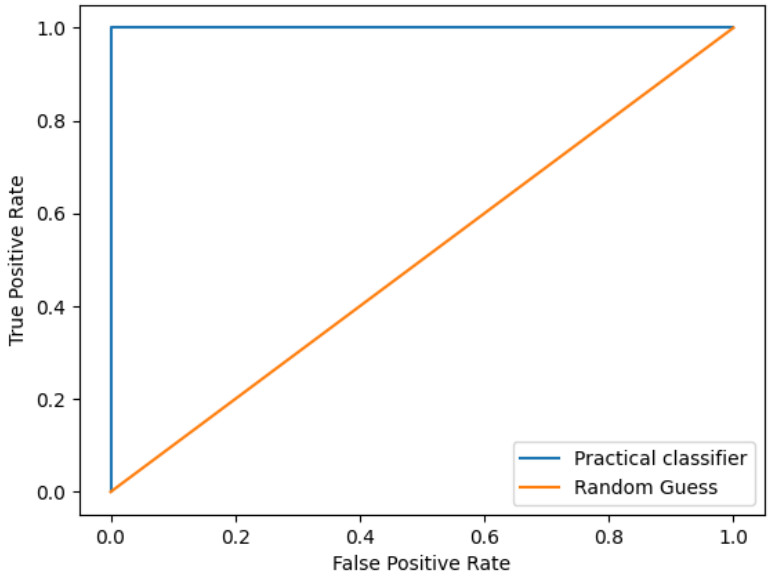
Матем. ожид. X1 (класс -1)	Матем. ожид. X2 (класс -1)	СКО (класс -1)	Матем. ожид. X1 (класс 1)	Матем. ожид. X2 (класс 1)	СКО (класс 1)	Количество элементов (класс -1)	Количество элементов (класс 1)
21	25	4	8	12	3	40	60

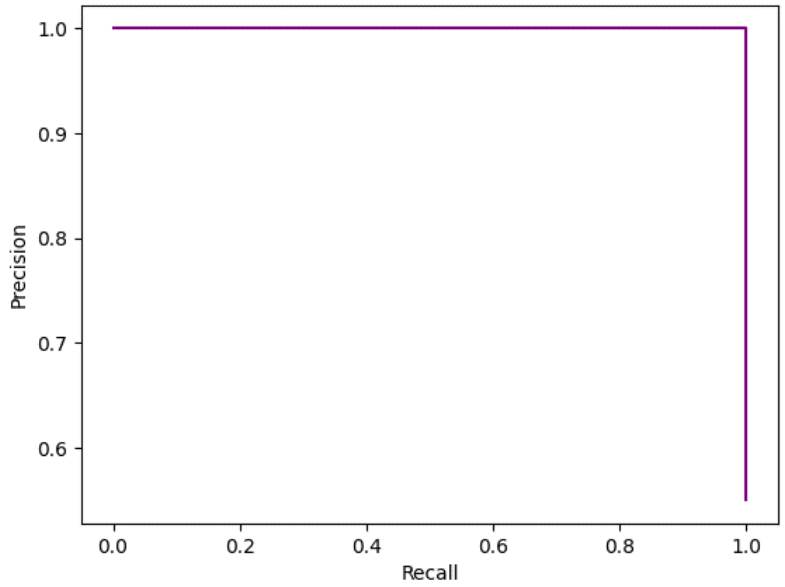
Построили диаграмму, иллюстрирующую данные (класс 1 – зеленые точки, класс -1 – красные)



Построили Байесовский классификатор, нашли точность, построили матрицу ошибок, ROC-кривую и PR-кривую:

```
Точность: 1.0
Матрица ошибок:
 [[ 9  0]
 [ 0 11]]
```



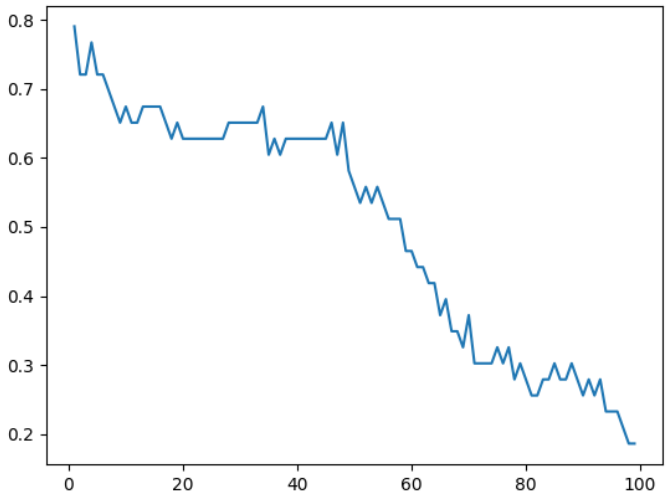


Получили идеальные точность, матрицу и кривые, отсюда делаем вывод, что построенный классификатор является «хорошим».

Задание 3

Построили классификатор на основе метода k ближайших соседей для обучающего множества Glass (glass.csv).

График зависимости точности классификации от количества ближайших соседей:



С увеличением количества ближайших соседей точность уменьшается
Для каждой метрики нашли, какую максимальную точность может дать классификатор, получилось, что максимальную точность дает метрика:

```
Оптимальная метрика: braycurtis
```

Все метрики: ["braycurtis",
"canberra",
"chebyshev",
"cityblock",
"correlation",
"cosine",
"euclidean",
"minkowski",
"sqeuclidean"]

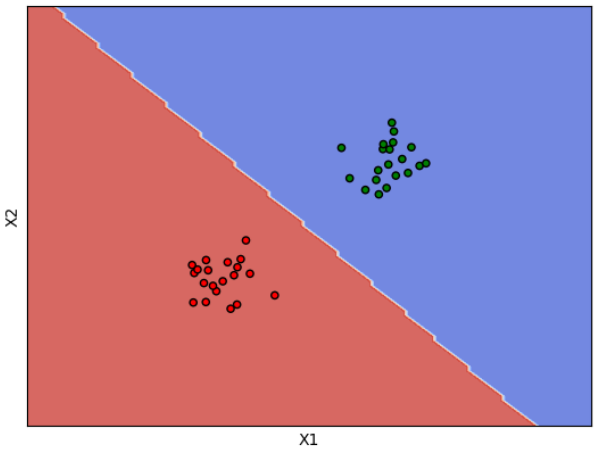
Экземпляр с характеристиками: RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1 относится к классу 5

```
[5]
```

Задание 4

Пункт а

Построили алгоритм метода опорных векторов с линейным ядром.
Визуализация данных:



Количество полученных опорных векторов для каждого класса равно 3

```
[3 3]
```

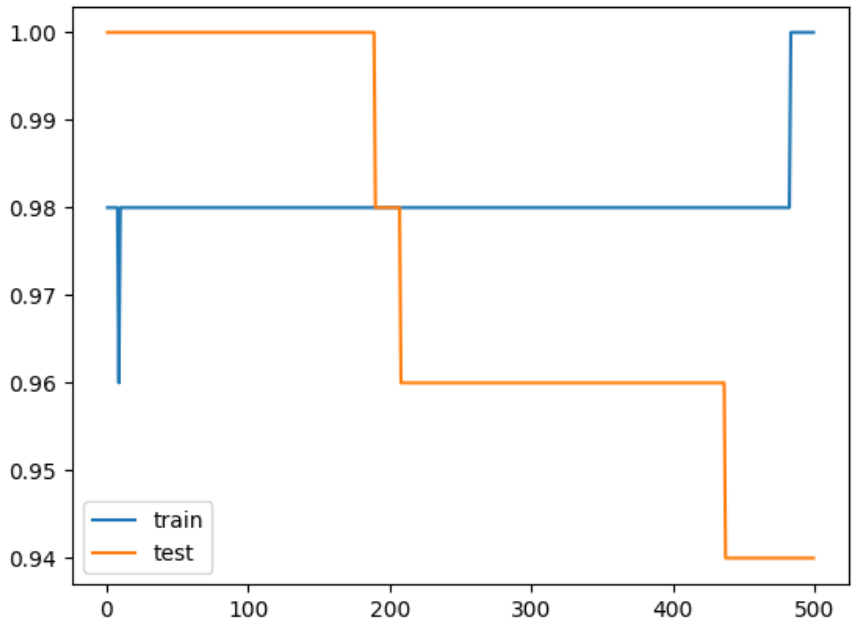
Матрицы ошибок классификации на обучающей и тестовой выборках:

```
Матрица ошибок для обучающей выборки:
[[20  0]
 [ 0 20]]
Матрица ошибок для тестовой выборки:
[[20  0]
 [ 0 20]]
```

Пункт b

Добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения штрафного параметра. Выберите оптимальное значение данного параметра и объясните свой выбор. Всегда ли нужно добиваться минимизации ошибки на обучающей выборке?

Изменяем штрафной параметр от 1 до 500:

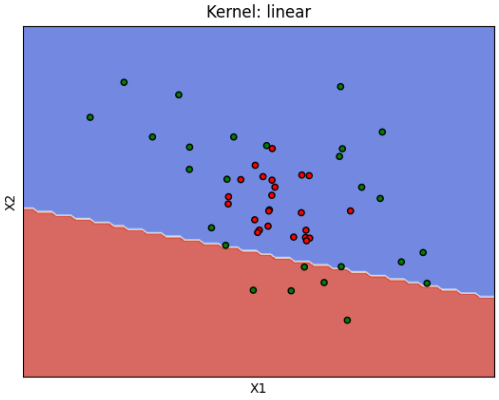


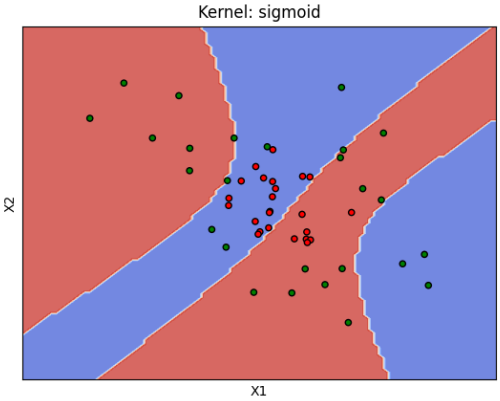
Точность становится максимальной на тренировочной выборке при C=483, однако на тестовой выборке точность снизилась, то есть модель переобучилась. На тестовой выборке начиная с C=190 точность снижается.

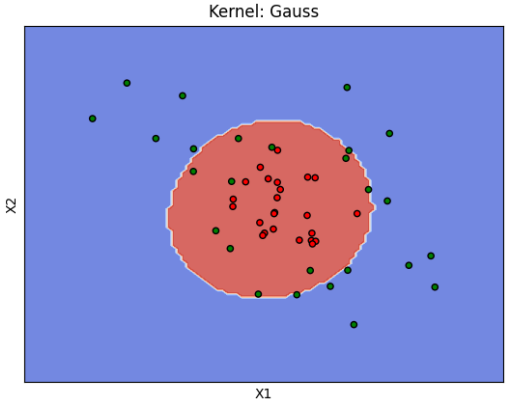
```
Test C =  190
Train C =  483
```

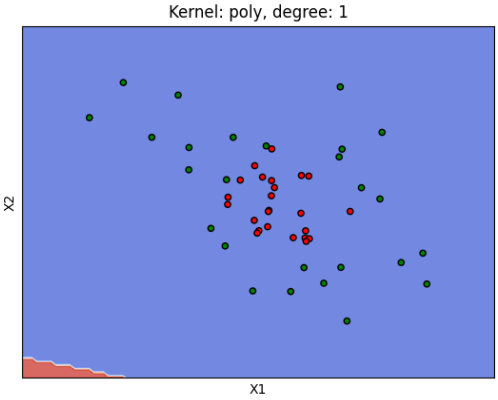
Не всегда нужно добиваться минимизации ошибки на обучающей выборке, так как это может привести к переобучению модели

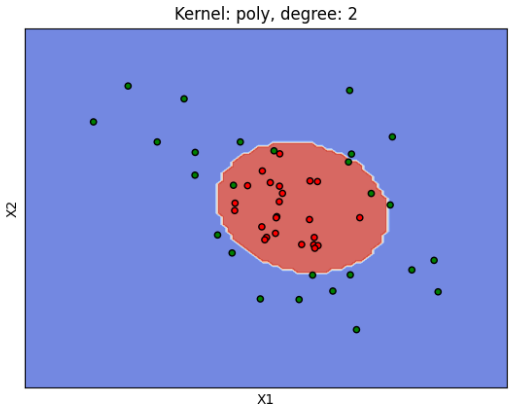
Пункт c

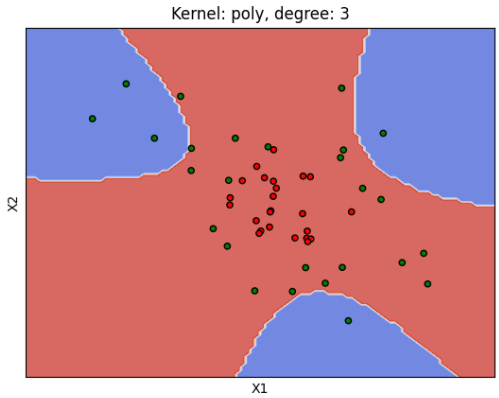

Kernel: linear
X2
X1

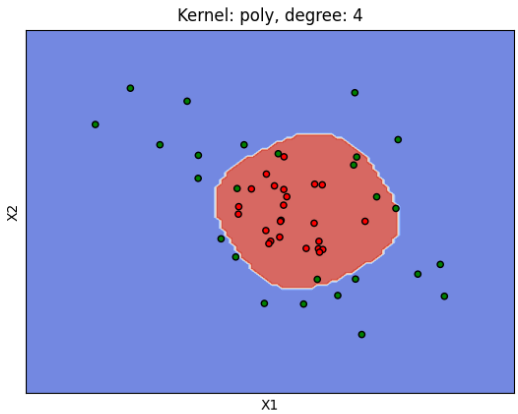

Kernel: sigmoid
X2
X1

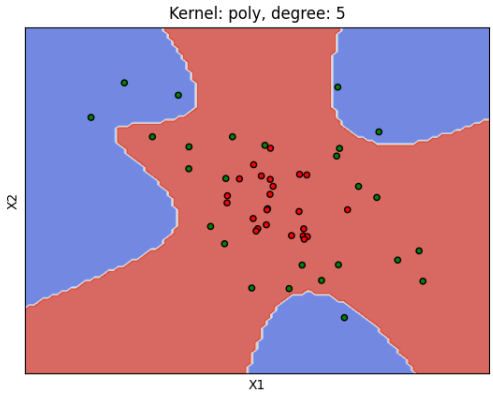

Kernel: Gauss
X2
X1


Kernel: poly, degree: 1
X2
X1


Kernel: poly, degree: 2
X2
X1

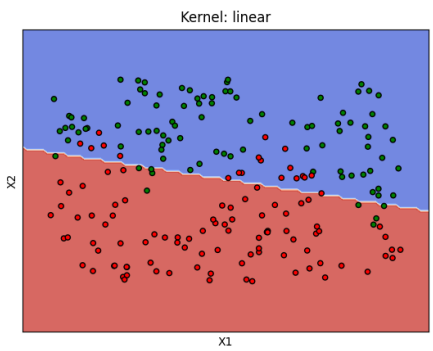

Kernel: poly, degree: 3
X2
X1

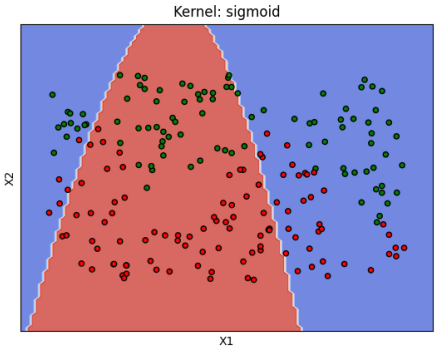

Kernel: poly, degree: 4
X2
X1

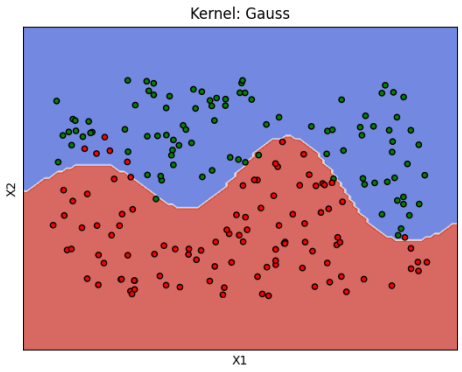

Kernel: poly, degree: 5
X2
X1

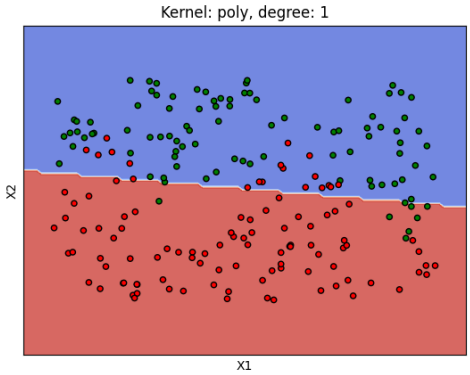
Линейное ядро, сигмоида и полиномиальные ядра нечетных степеней плохо классифицируют данные, лучше всего разделили данные полиномиальные 2 и 4 степеней и гауссово ядра.

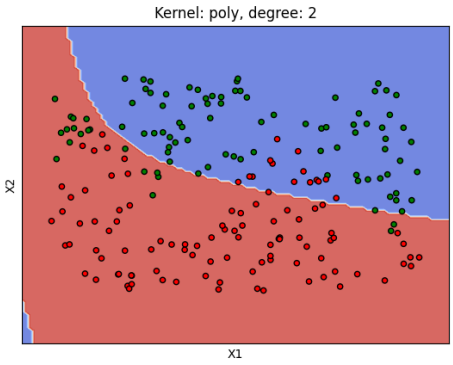
Пункт d

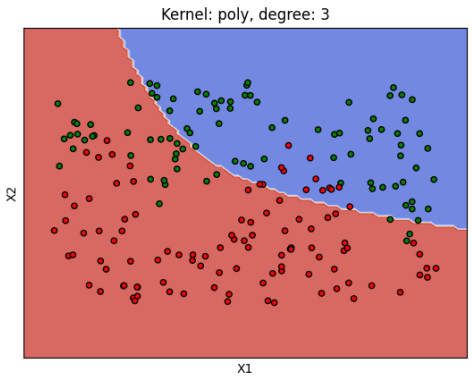


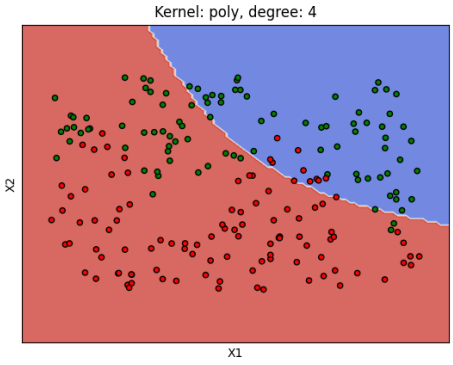


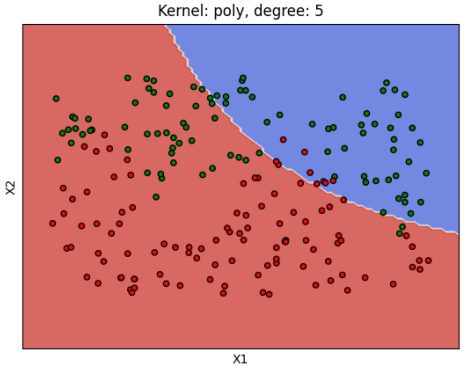






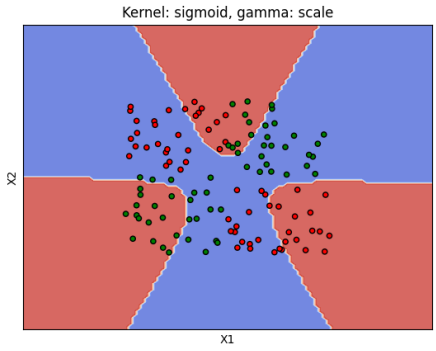


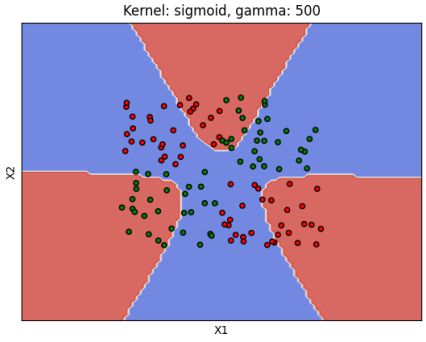


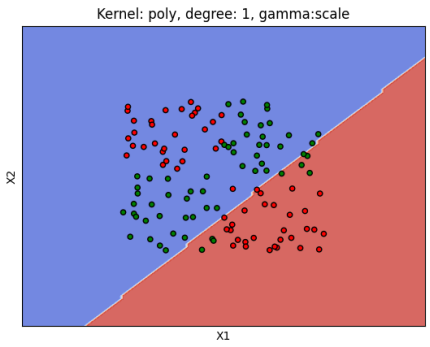


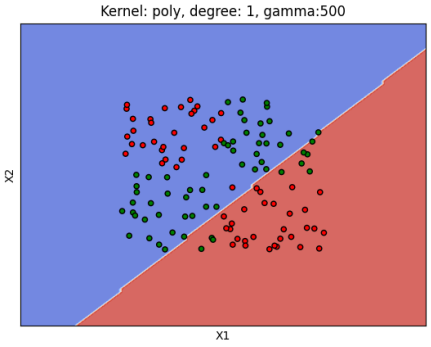
Хуже всего SVM классифицирует данные при использовании сигмоидальной функции, при повышении степени полинома качество классификации также ухудшается, лучше всего классификация проходит при применении Гауссова ядра

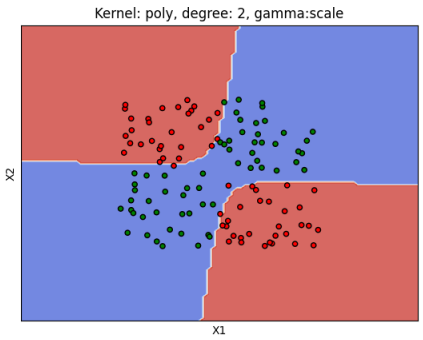
Пункт e

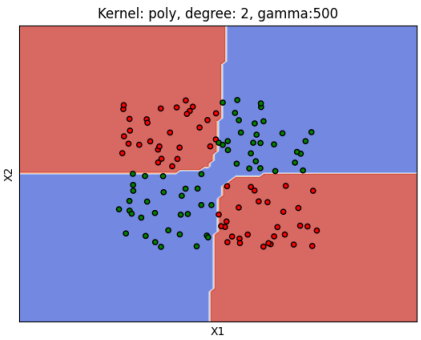




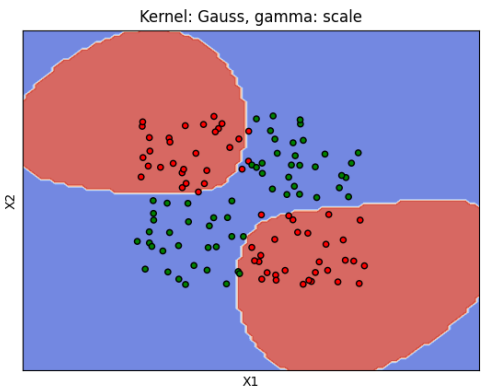


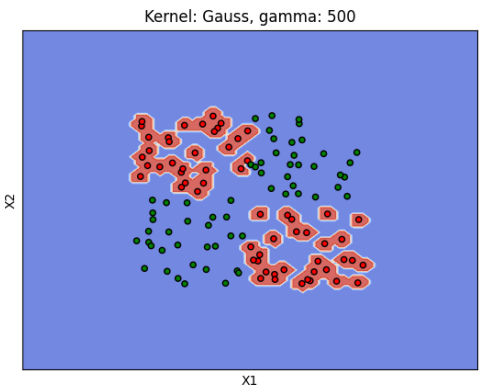






При использовании сигмоиды и полиномиальных ядер при увеличении параметра ядра переобучения не происходит





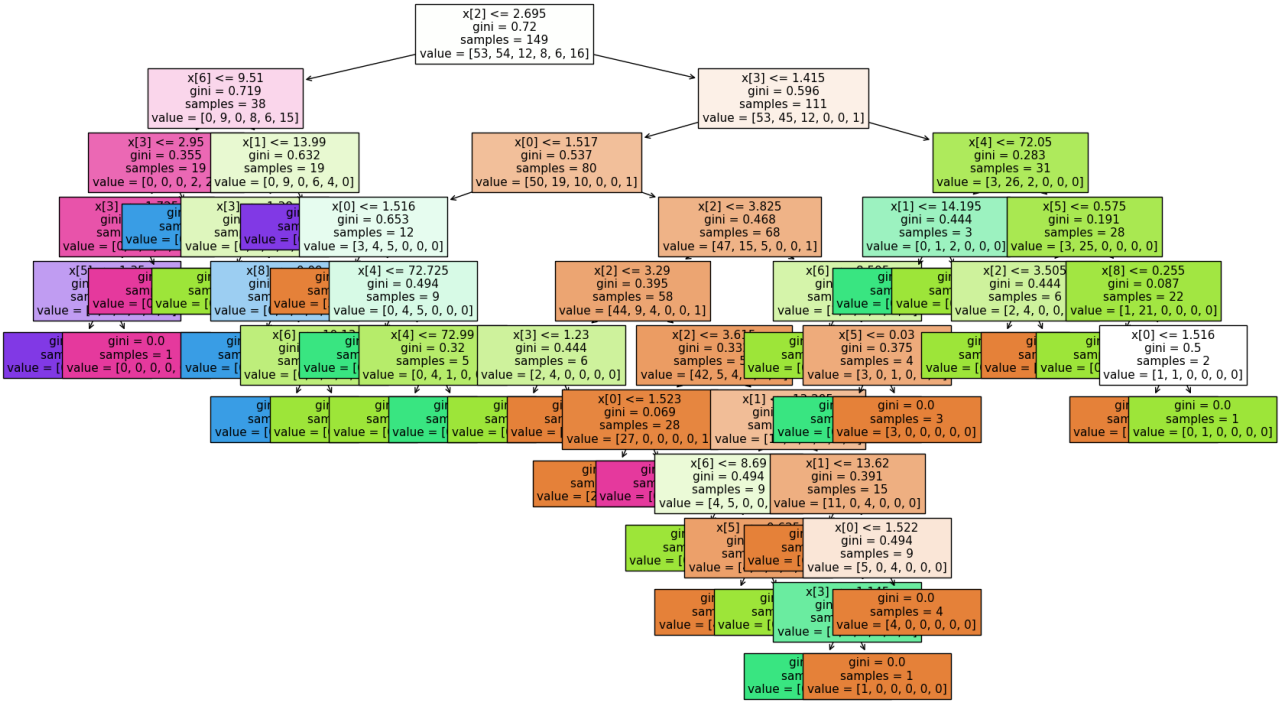
Увеличение параметра Гауссова ядра приводит к переобучению

Задание 5

Пункт а

Получили дерево глубиной 10

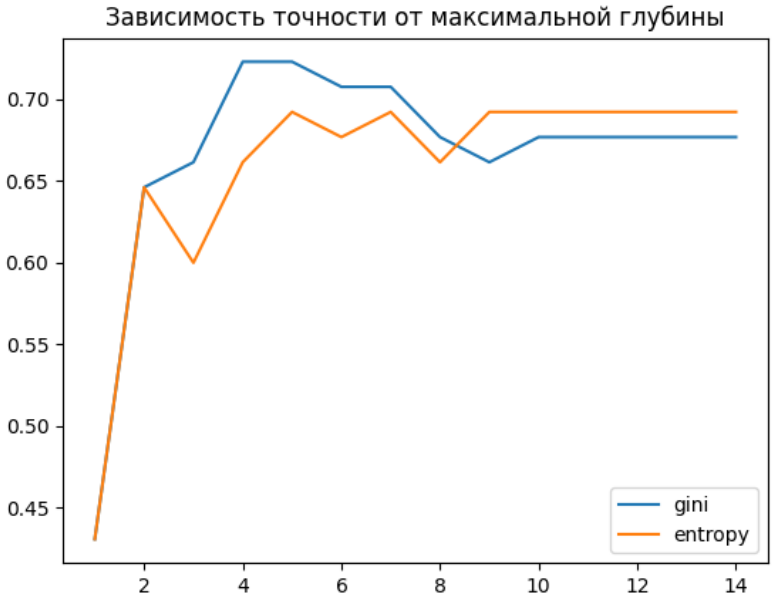
```
Accuracy without max_depth arg: 0.676923076923077
Tree depth: 10
```



Дерево получилось избыточным, так как с помощью дерева глубины 4, получаем точность классификации выше, чем у этого дерева

```
Accuracy with max_depth=4: 0.7230769230769231
Tree depth: 4
```

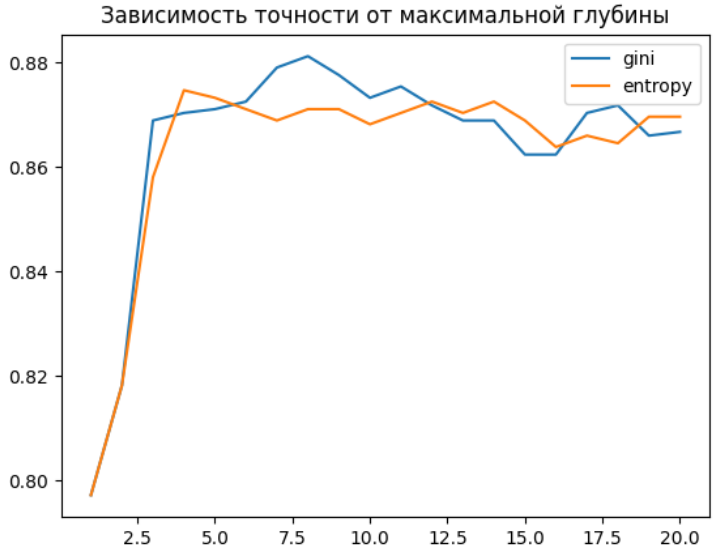
Исследовали зависимость точности классификации от критерия расщепления, максимальной глубины дерева



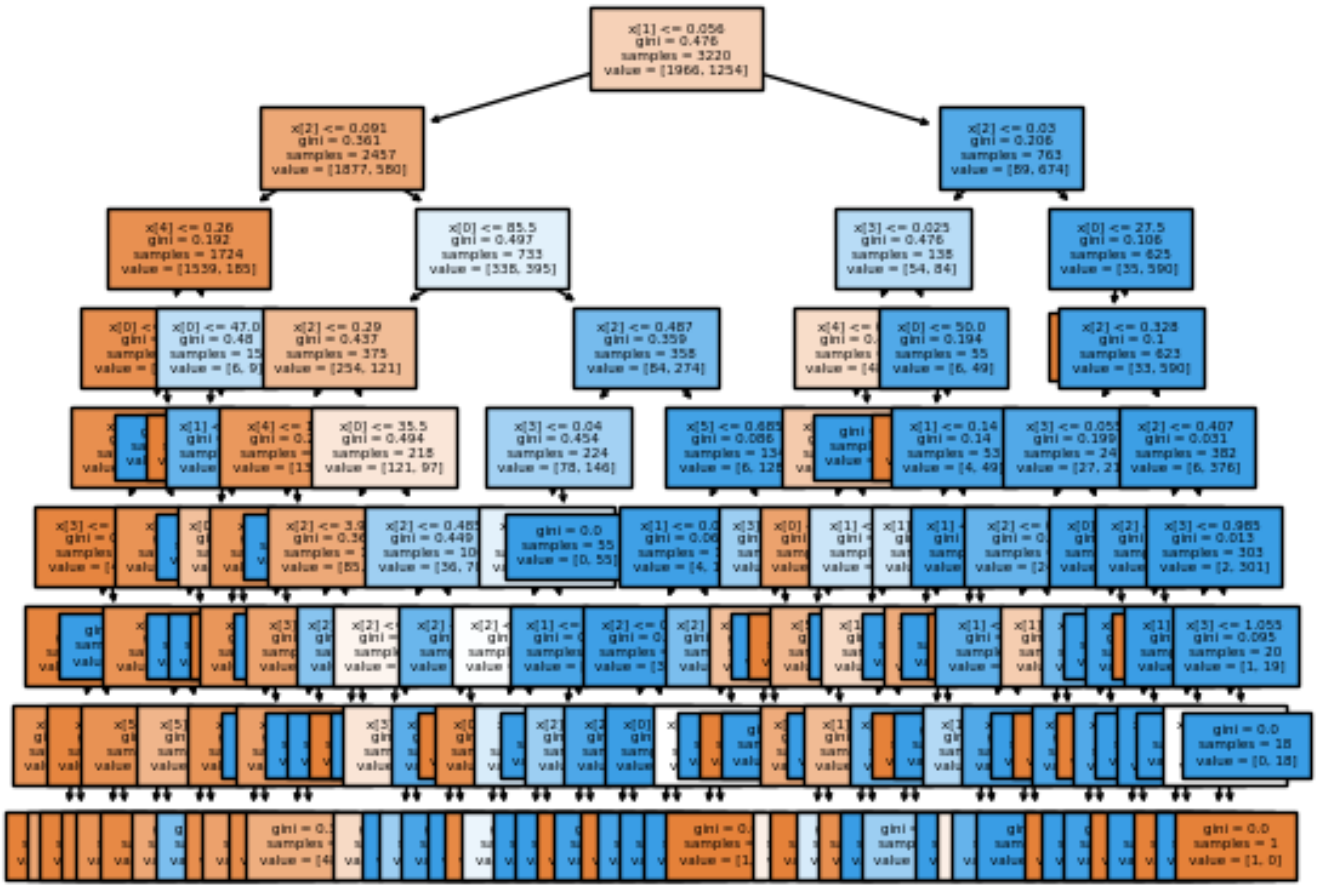
Для максимальной глубины дерева до 9 лучше себя показывает критерий Джинни, дальше для энтропии точность выше.
При максимальной глубине 4 мы получаем самую высокую точность, далее она снижается.

Пункт б

Для датасета spam7.csv получаем следующий график



Максимальная точность получается при использовании критерия Джинни с максимальной глубиной дерева = 8



Задание 6

Исходя из специфики задачи, наилучшим будет тот классификатор, который примет меньше всего решений о выдаче кредитов тем людям, у которых финансовая ситуация ухудшится, также лучше выдавать кредиты тем людям, которые могут их выплатить, чтобы не терять прибыль.

Получили следующие матрицы:

```
Матрица ошибок для наивного Байесовского классификатора:
46 64
1600 22343
Матрица ошибок для метода k ближайших соседей при k=1:
987 455
659 21952
Матрица ошибок для метода решающих деревьев при максимальной глубине 40 и критерии расщепления gini:
1117 613
529 21794
```

Используя наивный Байесовский классификатор, мы бы выдали больше всего кредитов людям, которые не смогут их выплатить.

Меньше всего кредитов людям, которые не смогут их выплатить, мы бы выдали, используя метод решающих деревьев, поэтому я считаю, что для данной ситуации лучше всего использовать именно этот классификатор.

Текст программы

ex1

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import pandas as pd
import matplotlib.pyplot as plt
from tabulate import tabulate


def load_txt(file_name):
    df = pd.read_csv(file_name, header=None)
    X = pd.get_dummies(df.iloc[:, :-1])
    Y = df.iloc[:, -1]
    return X, Y


def load_csv(file_name):
    df = pd.read_csv(file_name)
    X = df.iloc[:, :-1]
    Y = df.iloc[:, -1]
    return X, Y


def naive_bayes_result(X_digits, Y_digits, list_percent):
    dict_percent_accuracy = {}
    for percent in list_percent:
        X_train, X_test, Y_train, Y_test = train_test_split(X_digits,
Y_digits,

test_size=percent, random_state=123)
        gaussian_nb = GaussianNB()
        gaussian_nb.fit(X_train, Y_train)
        accuracy = gaussian_nb.score(X_test, Y_test)
        dict_percent_accuracy[percent] = accuracy
    return dict_percent_accuracy


X_tic_tac, Y_tic_tac = load_txt("tic_tac_toe.txt")
test_percent = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
tic_tac_result = naive_bayes_result(X_tic_tac, Y_tic_tac, test_percent)
X_spam, Y_spam = load_csv("spam.csv")
spam_result = naive_bayes_result(X_spam, Y_spam, test_percent)
print(tabulate(tic_tac_result.items(), headers=['Доля тестовых данных',
'Точность']))
print(tabulate(spam_result.items(), headers=['Доля тестовых данных',
'Точность']))

fig, axis = plt.subplots(1, 2)
fig.suptitle("Зависимость точности от доли тестовой части в выборке для:")
axis[0].plot(tic_tac_result.keys(), tic_tac_result.values(), color='red')
axis[0].set_ylim(0, 1)
axis[0].set_title('крестиков-ноликов')
axis[1].plot(spam_result.keys(), spam_result.values(), color='red')
axis[1].set_ylim(0, 1)
axis[1].set_title('спама')

plt.show()
```

ex2

```
from collections import namedtuple

import numpy as np
```

```
from matplotlib import pyplot as plt
from numpy import random
from sklearn import metrics
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

Points = namedtuple('Points', ['x', 'y'])
class_one = Points(random.normal(8, 3, 60), random.normal(12, 3, 60))
class_minus_one = Points(random.normal(21, 4, 40), random.normal(25, 4, 40))

plt.scatter(class_one.x, class_one.y, c='green')
plt.scatter(class_minus_one.x, class_minus_one.y, c='red')
plt.show()

X, Y = np.array(list(zip(class_one.x, class_one.y)) +
                list(zip(class_minus_one.x, class_minus_one.y))), \
    [1] * 60 + [-1] * 40
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.2,
random_state=123)
gaussian_nb = GaussianNB()
gaussian_nb.fit(X_train, Y_train)
accuracy = gaussian_nb.score(X_test, Y_test)
print("Точность:", accuracy)
Y_pred = gaussian_nb.predict(X_test)
matrix = metrics.confusion_matrix(Y_test, Y_pred)
print("Матрица ошибок:\n", matrix)
fpr, tpr, _ = metrics.roc_curve(Y_test, Y_pred)

# create ROC curve
plt.plot(fpr, tpr)
plt.plot([0, 1], [0, 1])
plt.legend(('Practical classifier', 'Random Guess'))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

precision, recall, thresholds = precision_recall_curve(Y_test, Y_pred)

#create precision recall curve
plt.plot(recall, precision, color='purple')

#add axis labels to plot
plt.ylabel('Precision')
plt.xlabel('Recall')

#display plot
plt.show()
```

ex3

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv("glass.csv")

X = pd.get_dummies(df.iloc[:, 1:-1])
Y = df.iloc[:, -1]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.2,
random_state=123)
```

```
all_metrics = ["braycurtis",
               "canberra",
               "chebyshev",
               "cityblock",
               "correlation",
               "cosine",
               "euclidean",
               "minkowski",
               "sqeuclidean"]
max_accuracy_metrics = {}
for metric in all_metrics:
    accuracy = []
    for n_neigh in range(1, 100):
        neigh = KNeighborsClassifier(n_neighbors=n_neigh, metric=metric)
        neigh.fit(X_train, Y_train)
        accuracy.append(neigh.score(X_test, Y_test))
    max_accuracy_metrics[metric] = max(accuracy)

max_key = max(max_accuracy_metrics, key=lambda k: max_accuracy_metrics[k])
print('Метрики и максимальные точности:', max_accuracy_metrics)
print('Оптимальная метрика:', max_key)
accuracy = []
for n_neigh in range(1, 100):
    neigh = KNeighborsClassifier(n_neighbors=n_neigh, metric=max_key)
    neigh.fit(X_train, Y_train)
    accuracy.append(neigh.score(X_test, Y_test))

print(accuracy)

plt.plot(range(1, 100), accuracy)
plt.show()

neigh = KNeighborsClassifier(n_neighbors=2, metric='correlation')
neigh.fit(X_train, Y_train)
Y_predict = neigh.predict([[1.516, 11.7, 1.01, 1.19, 72.59, 0.43, 11.44,
0.02, 0.1]])
print(Y_predict)
```

ex4

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import svm, metrics
from sklearn.inspection import DecisionBoundaryDisplay


def get_plot(X, classif, Y):
    fig, ax = plt.subplots()
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    X0, X1 = X.iloc[:, 0], X.iloc[:, 1]

    disp = DecisionBoundaryDisplay.from_estimator(
        classif,
        X,
        response_method="predict",
        cmap=plt.cm.coolwarm,
        alpha=0.8,
        ax=ax
    )
    ax.scatter(X0, X1, c=Y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
    ax.set_xticks(())
    ax.set_yticks(())
```

```
def read_txt(file_name):
    df = pd.read_csv(file_name, sep='\t')
    X = df.iloc[:, :-1]
    Y = df.iloc[:, -1]
    return X, Y


X_train, Y_train = read_txt('svmdata_a.txt')
X_test, Y_test = read_txt('svmdata_a_test.txt')
clf = svm.SVC(kernel='linear')
clf.fit(X_train, Y_train)
print(clf.n_support_)
Y_pred_test = clf.predict(X_test)
matrix_test = metrics.confusion_matrix(Y_test, Y_pred_test)
Y_pred_train = clf.predict(X_train)
matrix_train = metrics.confusion_matrix(Y_train, Y_pred_train)
print(f"Матрица ошибок для обучающей выборки:\n{matrix_train}\nМатрица ошибок
для тестовой выборки:\n{matrix_test}")
get_plot(X_train, clf, Y_train)
plt.show()

X_train, Y_train = read_txt('svmdata_b.txt')
X_test, Y_test = read_txt('svmdata_b_test.txt')
train_score = []
test_score = []
flag_test = False
flag_train = False
for param_c in range(1, 500):
    clf = svm.SVC(kernel='linear', C=param_c)
    clf.fit(X_train, Y_train)
    train_score.append(clf.score(X_train, Y_train))
    test_score.append(clf.score(X_test, Y_test))
    if train_score[-1] == 1 and flag_train is False:
        print("Train C = ", param_c)
        flag_train = True
    if test_score[-1] < 1 and flag_test is False:
        print("Test C = ", param_c)
        flag_test = True

plt.plot(range(1, 500), train_score)
plt.plot(range(1, 500), test_score)
plt.legend(('train', 'test'))
plt.show()


def different_kernels_visualize(X_train, Y_train):
    kernels = ['linear', 'sigmoid']

    for ker in kernels:
        svc = svm.SVC(kernel=ker)
        svc.fit(X_train, Y_train)
        get_plot(X_train, svc, Y_train)
        plt.title(f"Kernel: {ker}")
        plt.show()

    svc = svm.SVC(kernel="rbf", gamma=0.1)
    svc.fit(X_train, Y_train)
    get_plot(X_train, svc, Y_train)
    plt.title(f"Kernel: Gauss")
    plt.show()

    degrees = [1, 2, 3, 4, 5]
```

```
    for deg in degrees:
        svc = svm.SVC(kernel='poly', degree=deg)
        svc.fit(X_train, Y_train)
        get_plot(X_train, svc, Y_train)
        plt.title(f"Kernel: poly, degree: {deg}")
        plt.show()


X_train, Y_train = read_txt('svmdata_c.txt')
X_test, Y_test = read_txt('svmdata_c_test.txt')

# different_kernels_visualize(X_train, Y_train)

X_train, Y_train = read_txt('svmdata_d.txt')
X_test, Y_test = read_txt('svmdata_d_test.txt')
# different_kernels_visualize(X_train, Y_train)

X_train, Y_train = read_txt('svmdata_e.txt')
X_test, Y_test = read_txt('svmdata_e_test.txt')

gamma = ['scale', 500]

for gam in gamma:
    kernels = ['sigmoid']

    for ker in kernels:
        svc = svm.SVC(kernel=ker)
        svc.fit(X_train, Y_train)
        get_plot(X_train, svc, Y_train)
        plt.title(f"Kernel: {ker}, gamma: {gam}")
        plt.show()

    svc = svm.SVC(kernel="rbf", gamma=gam)
    svc.fit(X_train, Y_train)
    get_plot(X_train, svc, Y_train)
    plt.title(f"Kernel: Gauss, gamma: {gam}")
    plt.show()

    degrees = [1, 2, 3, 4, 5]
    for deg in degrees:
        svc = svm.SVC(kernel='poly', degree=deg, gamma=gam)
        svc.fit(X_train, Y_train)
        get_plot(X_train, svc, Y_train)
        plt.title(f"Kernel: poly, degree: {deg}, gamma:{gam}")
        plt.show()
```

ex5

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz


def accuracy_plot(X_train, Y_train, X_test, Y_test):
    criteria = ['gini', 'entropy']
    for crit in criteria:
        accuracies = []
        for i in range(1, 51):
            tree_clf = DecisionTreeClassifier(random_state=0, max_depth=i,
criterion=crit)
            tree_clf.fit(X_train, Y_train)
            accuracies.append(tree_clf.score(X_test, Y_test))
        plt.plot(range(1, 51), accuracies)
```

```
    plt.title("Зависимость точности от максимальной глубины")
    plt.legend(('gini', 'entropy'))
    plt.show()


df = pd.read_csv("glass.csv")

X = pd.get_dummies(df.iloc[:, 1:-1])
Y = df.iloc[:, -1]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.3,
random_state=123)

clf = DecisionTreeClassifier(random_state=0)
clf.fit(X_train, Y_train)
print("Accuracy without max_depth arg:", clf.score(X_test, Y_test), "\nTree
depth:", clf.get_depth())

fig = plt.figure(figsize=(20, 12))
_ = tree.plot_tree(clf,
                   filled=True, fontsize=11)
plt.show()

clf2 = DecisionTreeClassifier(random_state=0, max_depth=4)
clf2.fit(X_train, Y_train)
print("Accuracy with max_depth=4:", clf2.score(X_test, Y_test), "\nTree
depth:", clf2.get_depth())

accuracy_plot(X_train, Y_train, X_test, Y_test)

df = pd.read_csv("spam7.csv")

X = df.iloc[:, :-1]
Y = df.iloc[:, -1]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.3,
random_state=123)

accuracy_plot(X_train, Y_train, X_test, Y_test)
tree_clf = DecisionTreeClassifier(random_state=0, max_depth=8)
tree_clf.fit(X_train, Y_train)
print(tree_clf.score(X_test, Y_test))
_ = tree.plot_tree(tree_clf,
                   filled=True, fontsize=4)
plt.show()
```

ex6

```
import pandas as pd
from sklearn import metrics, svm
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

from lab1.ex5 import accuracy_plot

df_train = pd.read_csv("bank_scoring_train.csv", sep='\t')

X_train = df_train.iloc[:, 1:]
Y_train = df_train.iloc[:, 0]

df_test = pd.read_csv("bank_scoring_test.csv", sep='\t')
```

```
X_test = df_test.iloc[:, 1:]
Y_test = df_test.iloc[:, 0]

gaussian_nb = GaussianNB()
gaussian_nb.fit(X_train, Y_train)

Y_pred = gaussian_nb.predict(X_test)

print("Матрица ошибок для наивного Байесовского классификатора:")
tn, fp, fn, tp = metrics.confusion_matrix(Y_test, Y_pred).ravel()
print(tp, fp)
print(fn, tn)

# accuracy = {}
# for n_neigh in range(1, 100):
#     neigh = KNeighborsClassifier(n_neighbors=n_neigh)
#     neigh.fit(X_train, Y_train)
#     accuracy[n_neigh] = neigh.score(X_test, Y_test)
#
# max_key = max(accuracy, key=lambda k: accuracy[k])
neigh = KNeighborsClassifier(n_neighbors=1)
neigh.fit(X_train, Y_train)
Y_pred = neigh.predict(X_test)

print(f"Матрица ошибок для метода k ближайших соседей при k={1}:")
tn, fp, fn, tp = metrics.confusion_matrix(Y_test, Y_pred).ravel()
print(tp, fp)
print(fn, tn)

# kernels = ['linear', 'sigmoid']
#
# for ker in kernels:
#     svc = svm.SVC(kernel=ker)
#     svc.fit(X_train, Y_train)
#     Y_pred = svc.predict(X_test)
#     print(f"Матрица ошибок для метода SVM при kernel={ker}:")
#     tn, fp, fn, tp = metrics.confusion_matrix(Y_test, Y_pred).ravel()
#     print(tp, fp)
#     print(fn, tn)

# svc = svm.SVC(kernel="rbf", gamma=0.1)
# svc.fit(X_train, Y_train)
# Y_pred = svc.predict(X_test)
# print(f"Матрица ошибок для метода SVM при kernel=rbf:")
# tn, fp, fn, tp = metrics.confusion_matrix(Y_test, Y_pred).ravel()
# print(tp, fp)
# print(fn, tn)
#
# degrees = [2, 3]
# for deg in degrees:
#     svc = svm.SVC(kernel='poly', degree=deg)
#     svc.fit(X_train, Y_train)
#     Y_pred = svc.predict(X_test)
#     print(f"Матрица ошибок для метода SVM при kernel=poly степени {deg}:")
#     tn, fp, fn, tp = metrics.confusion_matrix(Y_test, Y_pred).ravel()
#     print(tp, fp)
#     print(fn, tn)

# accuracy_plot(X_train, Y_train, X_test, Y_test)
tree_clf = DecisionTreeClassifier(random_state=0, max_depth=34)
tree_clf.fit(X_train, Y_train)
Y_pred = tree_clf.predict(X_test)
```

```
print(f"Матрица ошибок для метода решающих деревьев при максимальной глубине
40 и критерии расщепления gini:")
tn, fp, fn, tp = metrics.confusion_matrix(Y_test, Y_pred).ravel()
print(tp, fp)
print(fn, tn)
```