ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
ВШ программной инженерии

ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

Отчет по лабораторной работе № 1.

по дисциплине “Системы управления базами данных”

Выполнила
студентка гр. 3530202/00201 Козлова Е. А.

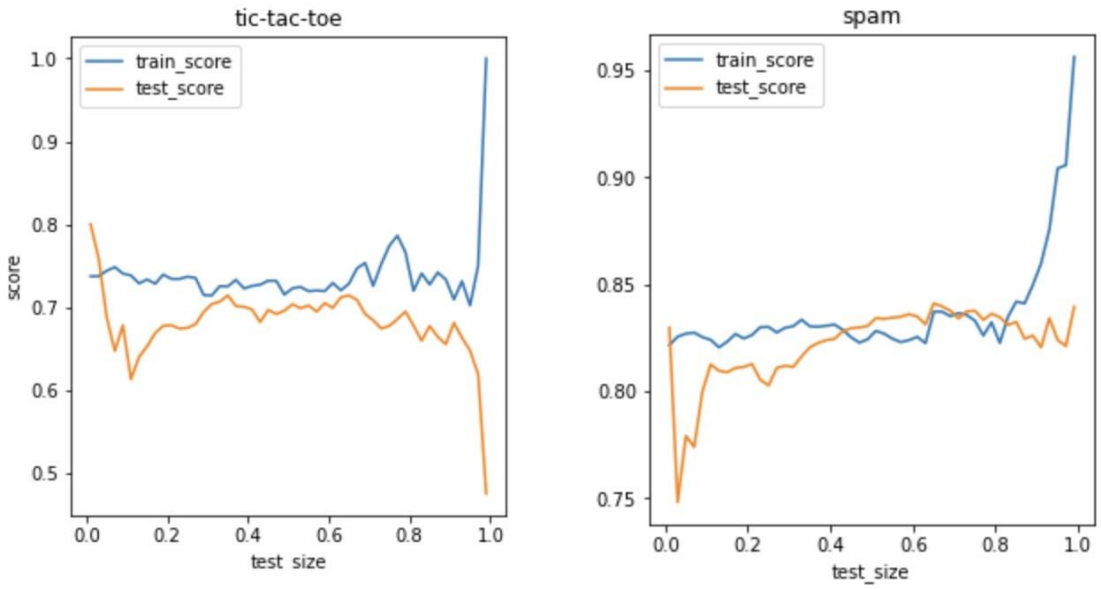
Руководитель Селин И. А.

Санкт-Петербург
2023

Оглавление

Задание 1.

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.



Задание 2.

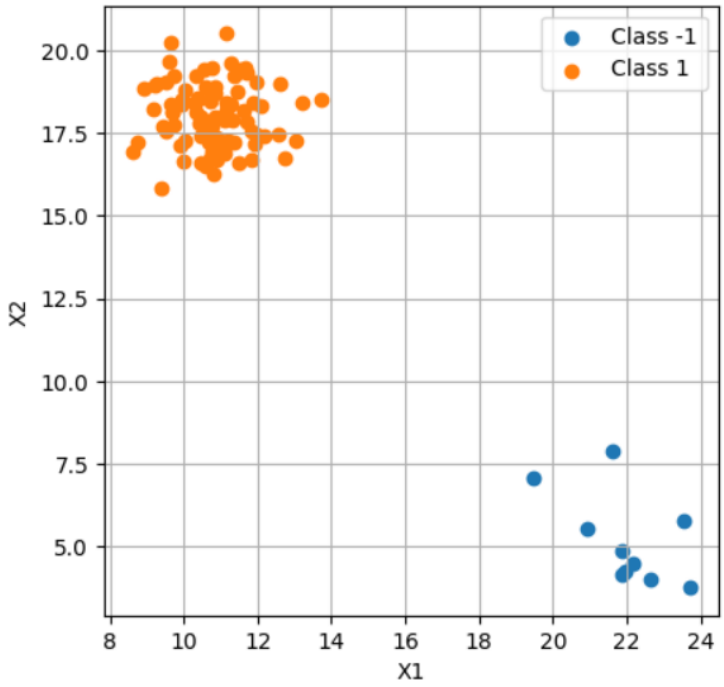
Вариант 10.

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»?

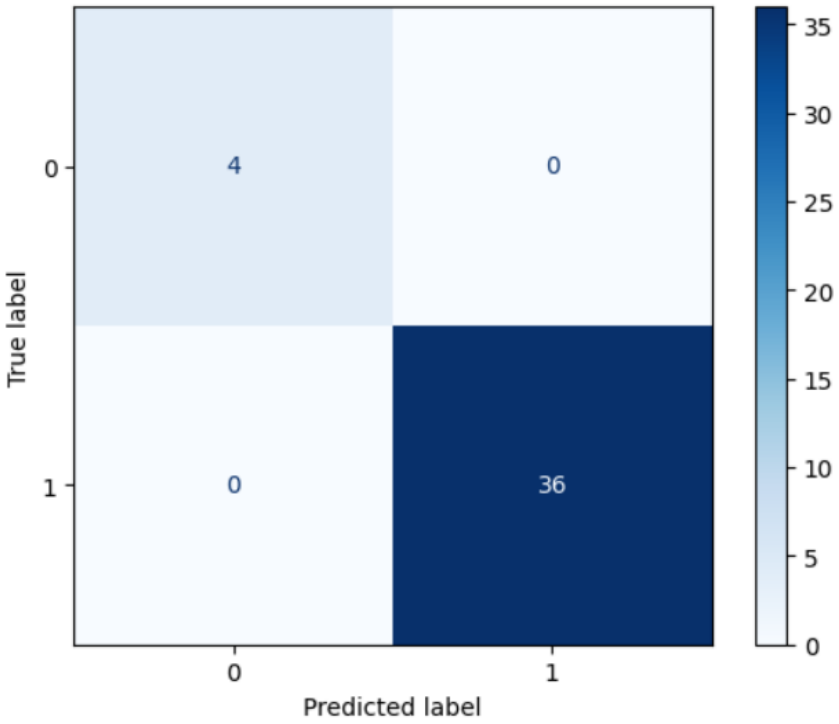
Вариант	Фамилия	Имя	Матем. ожид. X1 (класс -1)	Матем. ожид. X2 (класс -1)	СКО (класс -1)	Матем. ожид. X1 (класс 1)	Матем. ожид. X2 (класс 1)	СКО (класс 1)	Количество элементов (класс -1)	Количество элементов (класс 1)
10	Козлова	Елена	22	5	3	11	18	1	10	90

Точность классификатора: `1.0`

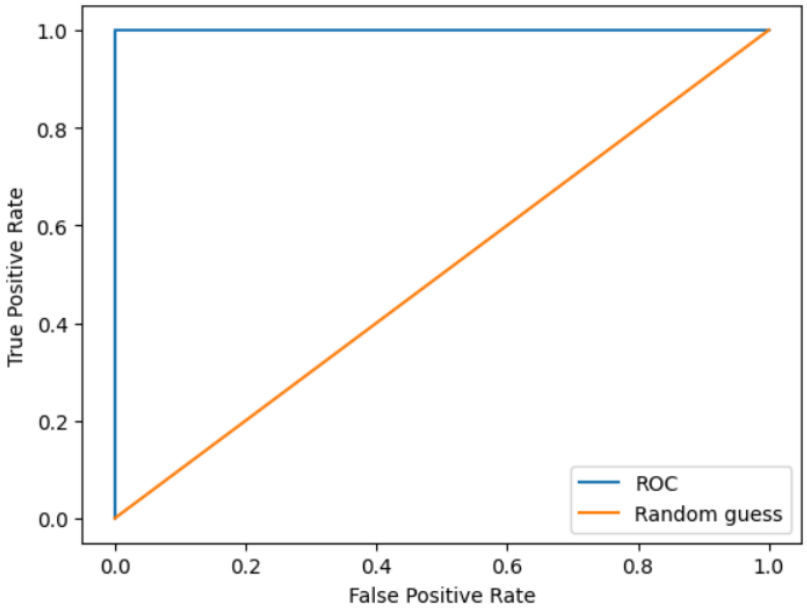
Диаграмма:



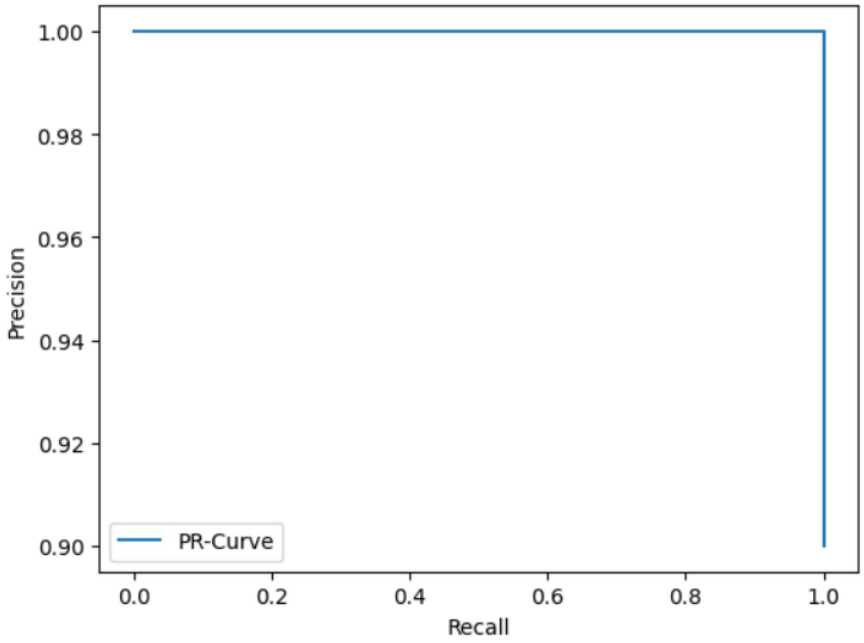
Матрица ошибок:



ROC-кривая:



PR-кривая:



Вывод:

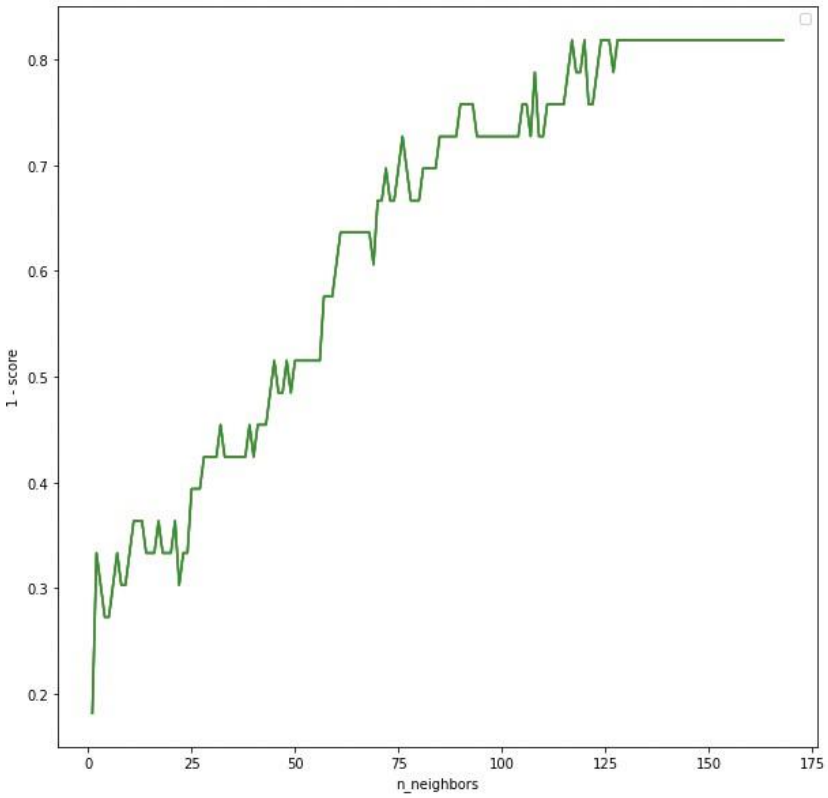
Достигнута высокая точность классификатора, что свидетельствует о его потенциале для успешного решения более сложных задач, как продемонстрировано в ходе выполнения данной лабораторной работы.

Задание 3.

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

График зависимости ошибки классификации от количества ближайших соседей:



Лучшая метрика:

```
grid_search_cv_knc.fit(X, y)
grid_search_cv_knc.best_params_
```

```
{'metric': 'euclidean', 'n_neighbors': 1}
```

Точность метрики:

```
grid_search_cv_knc.best_score_
```

```
0.640531561461794
```

Тип стекла:

```
grid_search_cv_knc.best_estimator_.predict(pd.DataFrame(data={'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]}))[0]
```

```
5
```

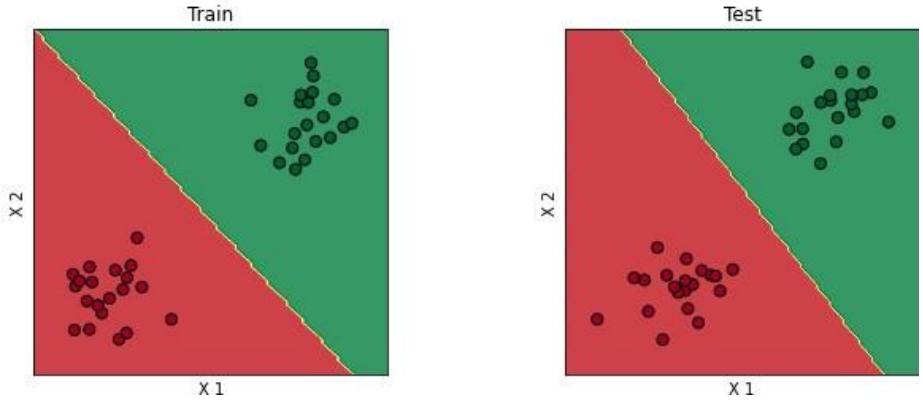
Задание 4.

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.

Модель опорных векторов с линейным ядром:

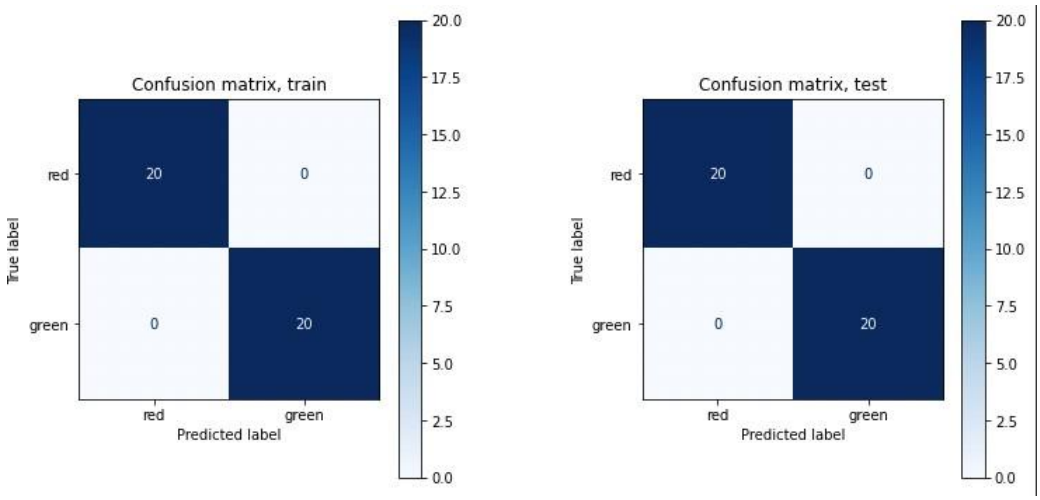


Количество полученных опорных векторов:

```
print("Support vectors number \nclass 'red':", svc.n_support_[0],
      "\nclass 'green':", svc.n_support_[1])
```

```
Support vectors number
class 'red': 3
class 'green': 3
```

Матрица ошибок классификации на обучающей и тестовой выборках:



Модель метода опорных векторов с линейным ядром:

```
print('train_score =', opt_score[0], ', C =', opt_C[0],
      '\ntest_score =', opt_score[1], ', C =', opt_C[1])
```

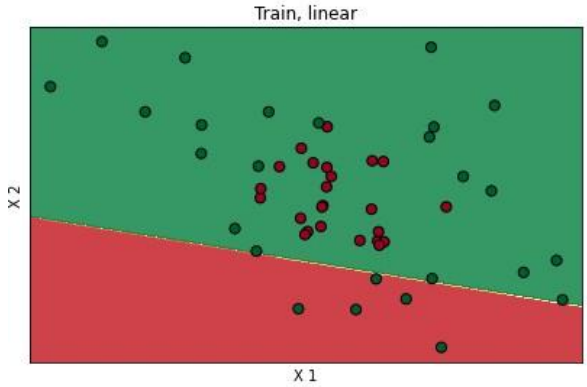
```
train_score = 1.0 , C = 483
test_score = 1.0 , C = 1
```

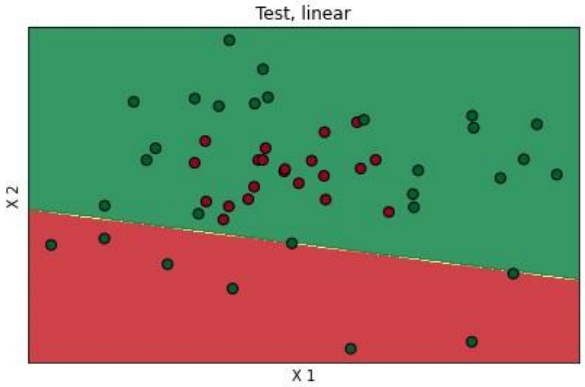
Оптимальное значение:

```
svc = SVC(kernel='linear')
grid_search_cv_clf = GridSearchCV(svc, {'C': range(1, 1000)}, cv=5, n_jobs=-1)
grid_search_cv_clf.fit(b_X_train, b_y_train)
grid_search_cv_clf.best_params_
```

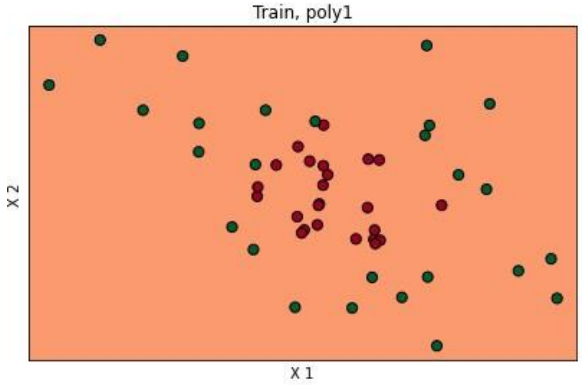
```
{'C': 1}
```

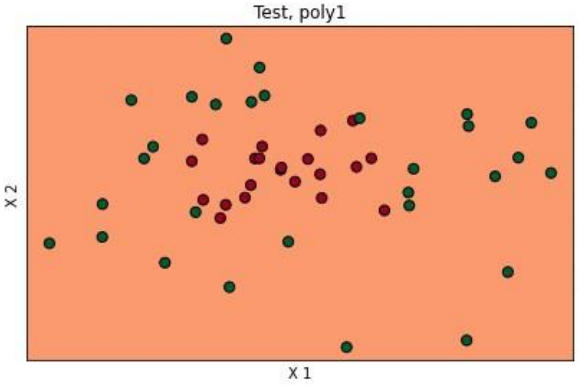
Модель метода опорных векторов с различными ядрами.
Линейное ядро:



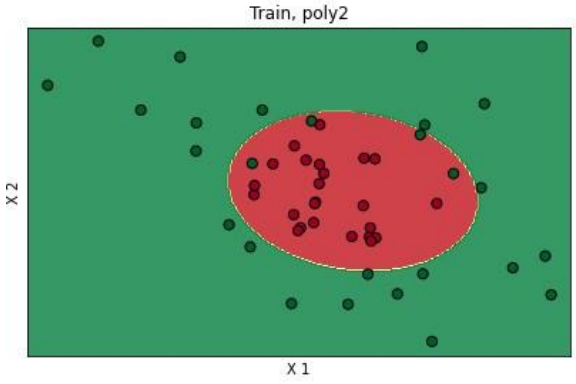


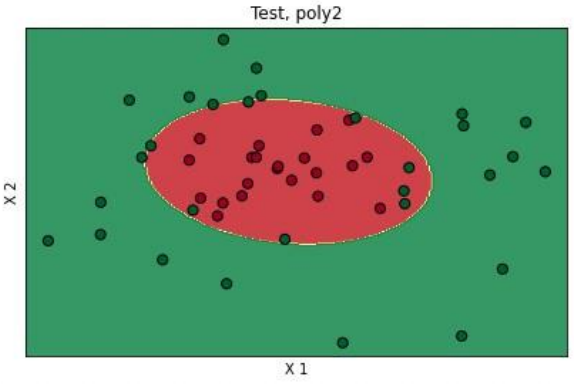
Полиномиальное ядро, степень 1:



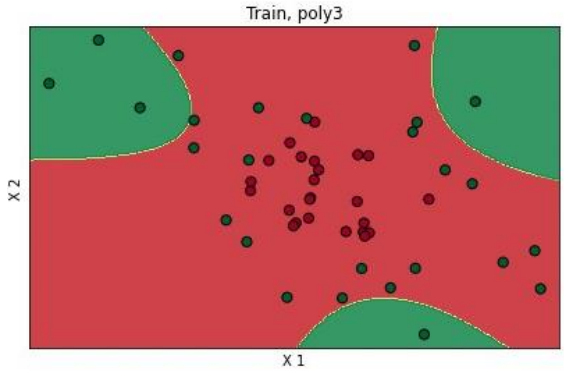


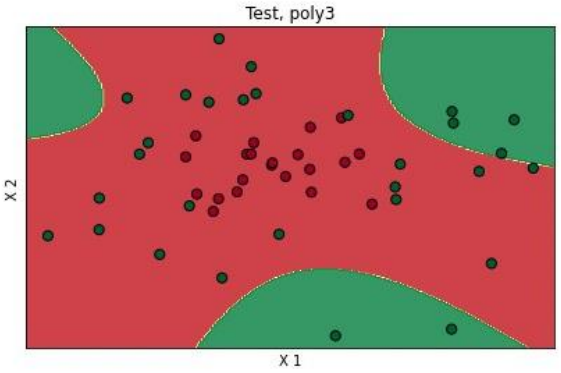
Полиномиальное ядро, степень 2:



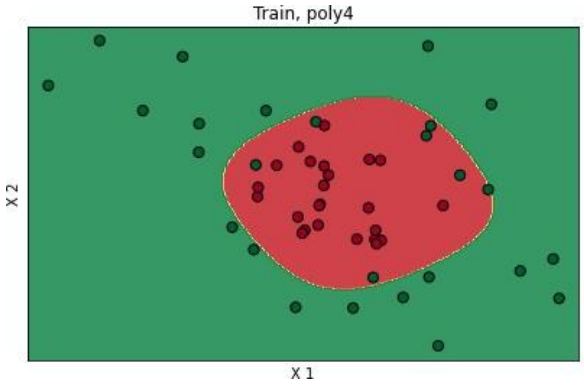


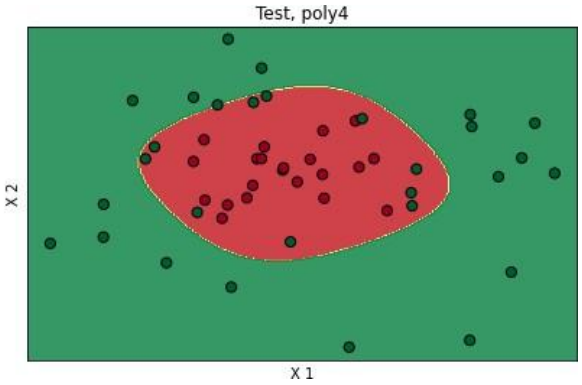
Полиномиальное ядро, степень 3:



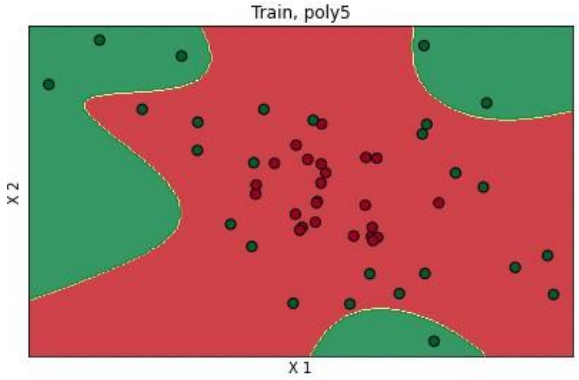


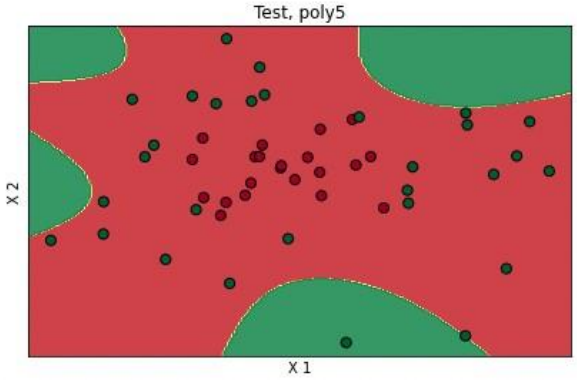
Полиномиальное ядро, степень 4:



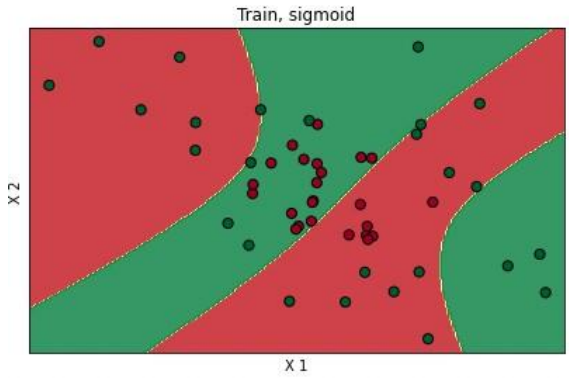


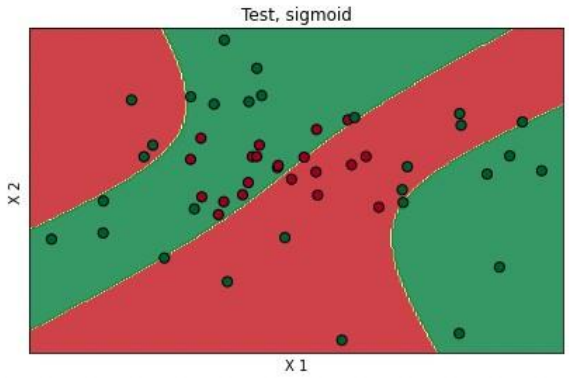
Полиномиальное ядро, степень 5:



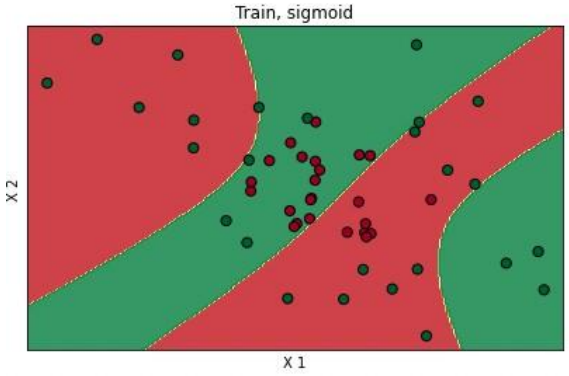


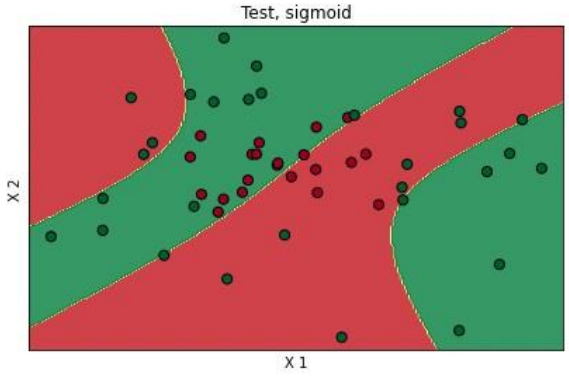
Сигмоидальная функция:



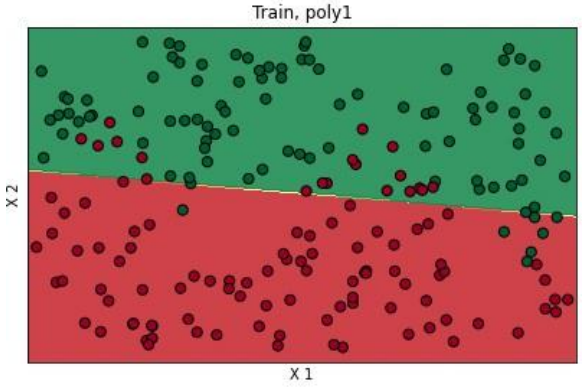


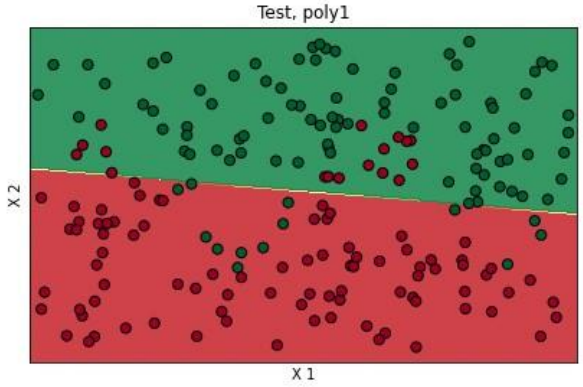
Гауссово:



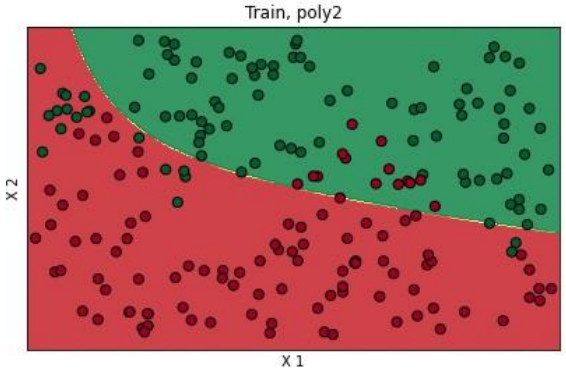


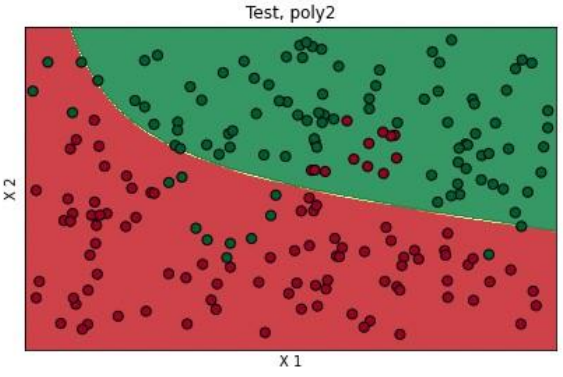
Модель метода опорных векторов с различными ядрами (разбиение пространства).
Полиномиальное ядро, степень 1:



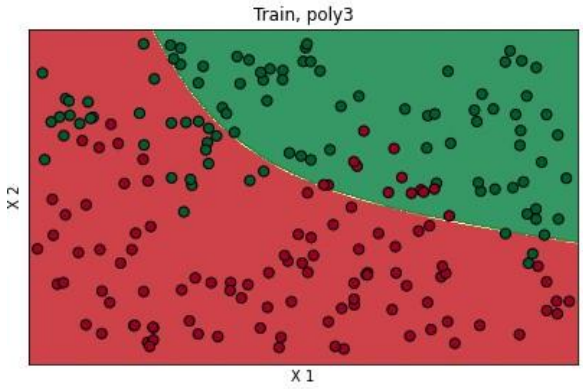


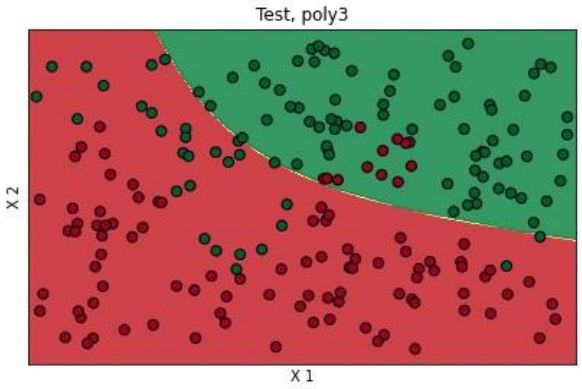
Полиномиальное ядро, степень 2:



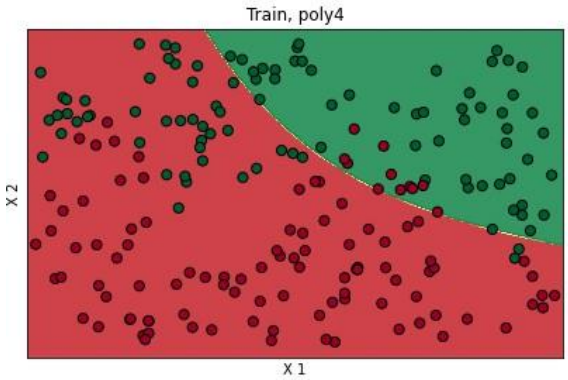


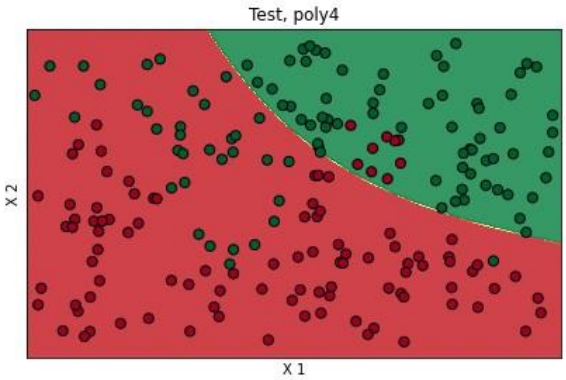
Полиномиальное ядро, степень 3:



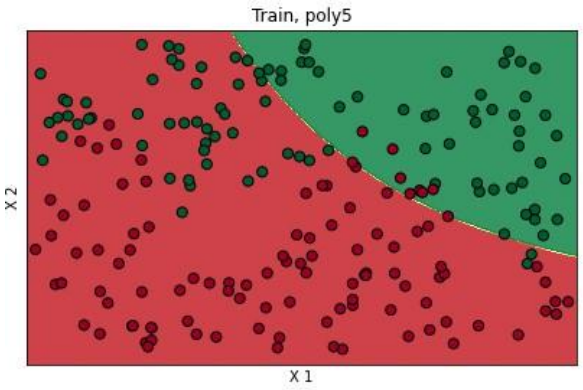


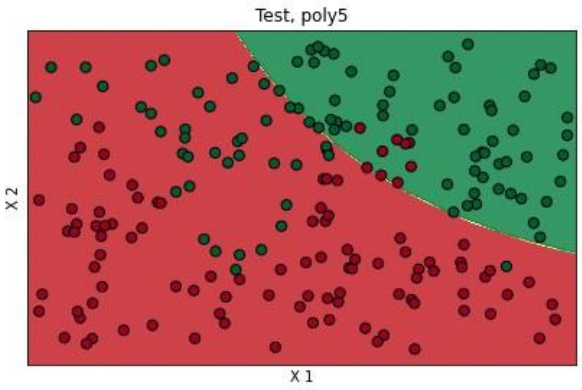
Полиномиальное ядро, степень 4:



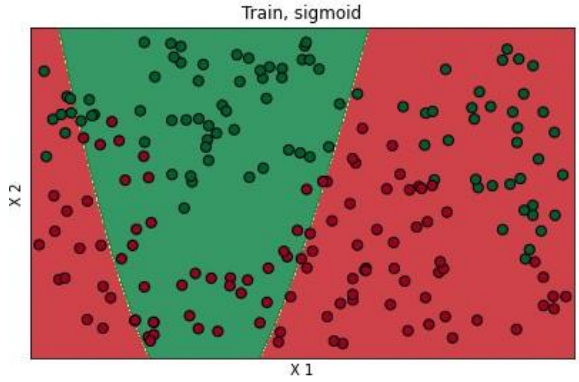


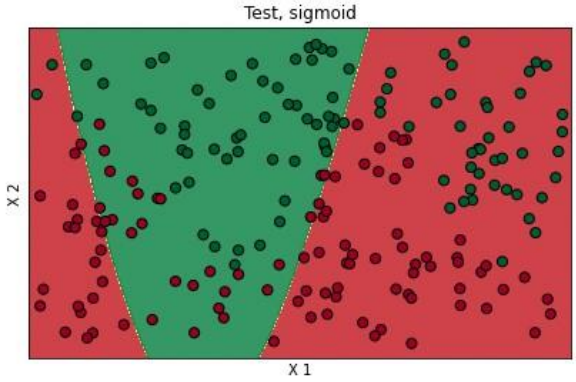
Полиномиальное ядро, степень 5:



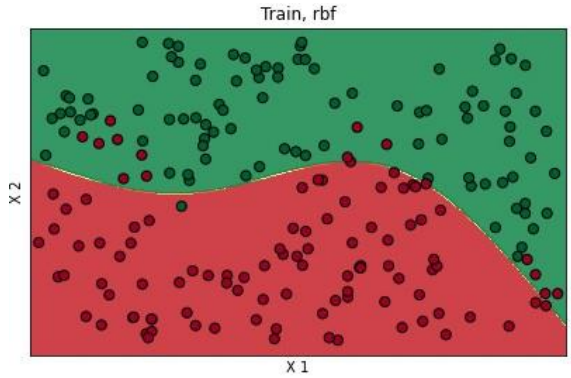


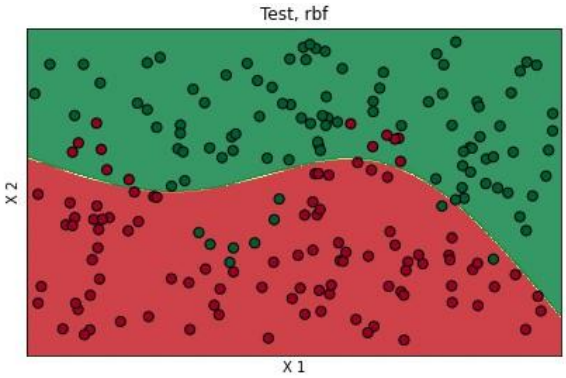
Сигмоидальная функция:



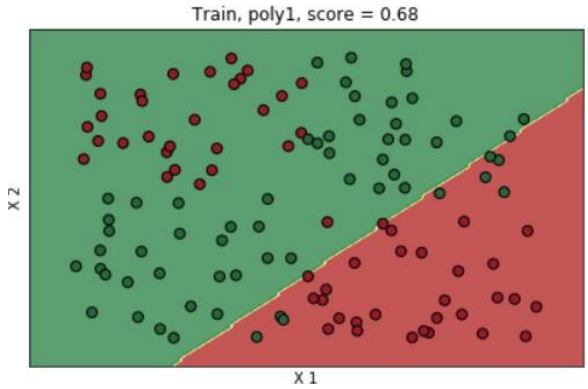


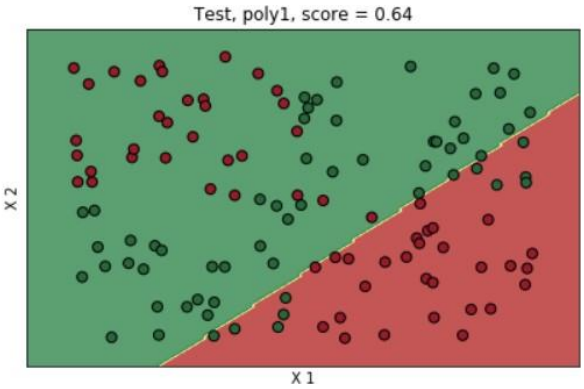
Гауссово:



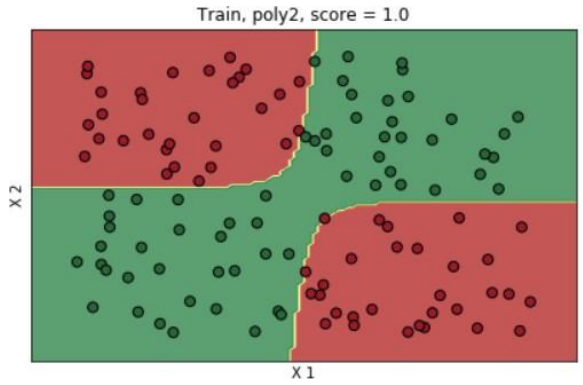


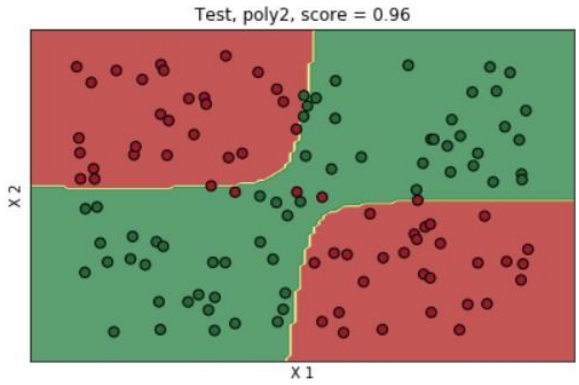
Модель метода опорных векторов с различными ядрами (эффект переобучения).
Переобученная модель, полиномиальное ядро, степень 1:



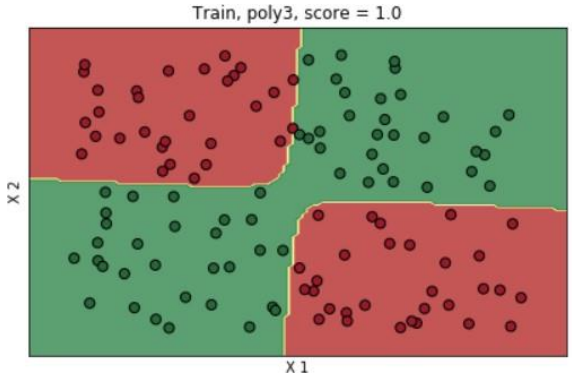


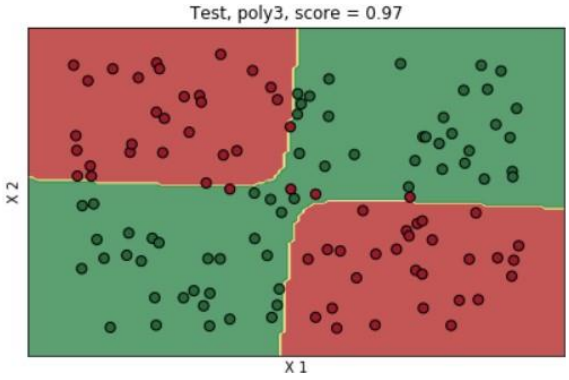
Переобученная модель, полиномиальное ядро, степень 2:



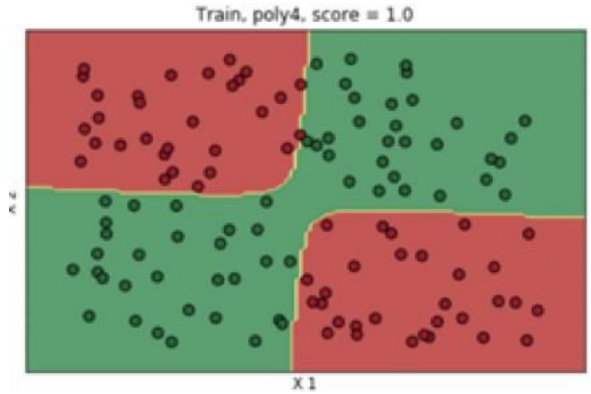


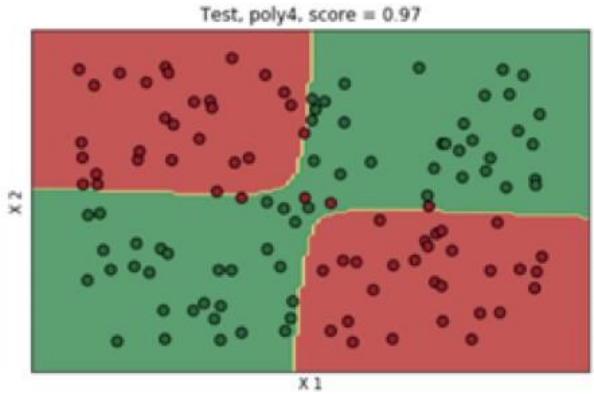
Переобученная модель, полиномиальное ядро, степень 3:



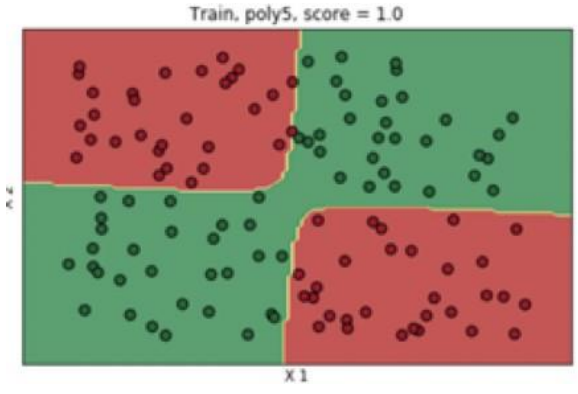


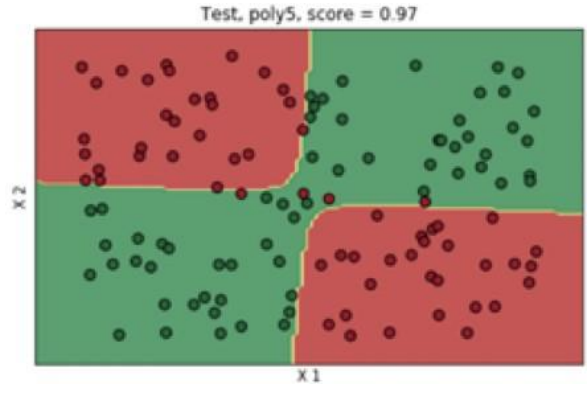
Переобученная модель, полиномиальное ядро, степень 4:



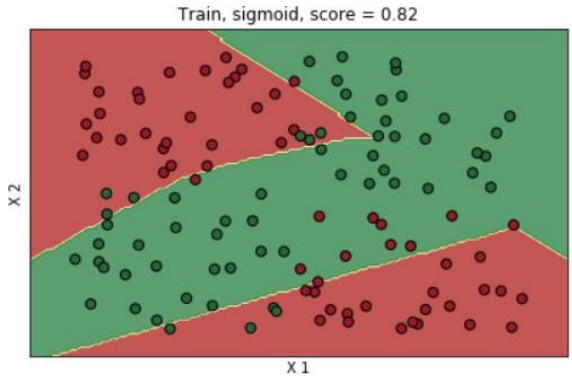


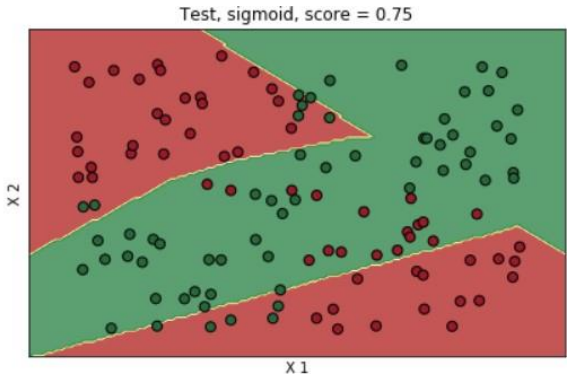
Переобученная модель, полиномиальное ядро, степень 5:



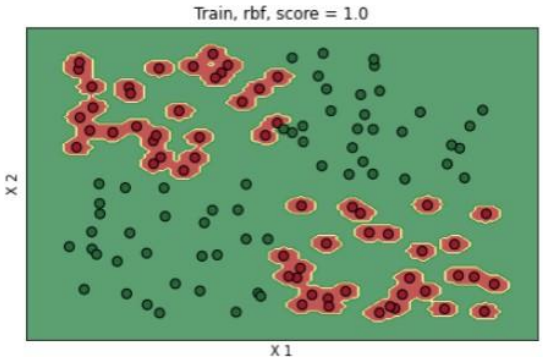


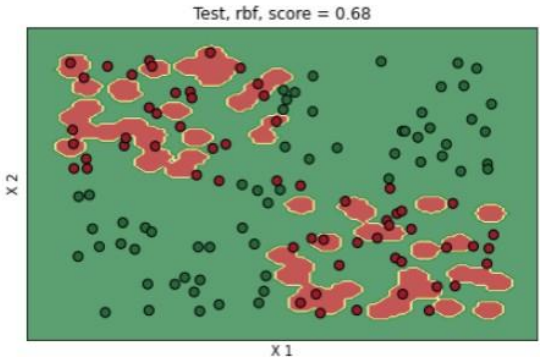
Сигмоидальная функция:





Гауссово:





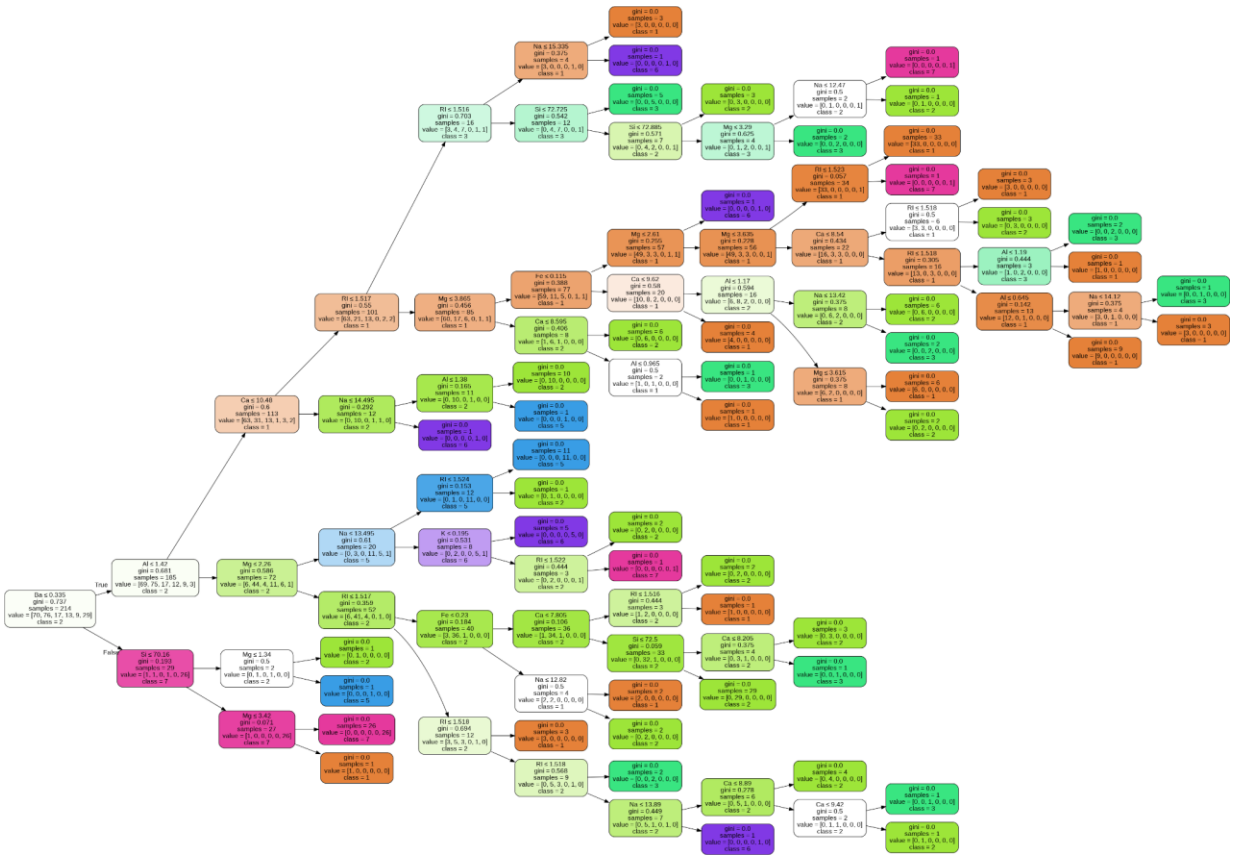
Задание 5.

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.

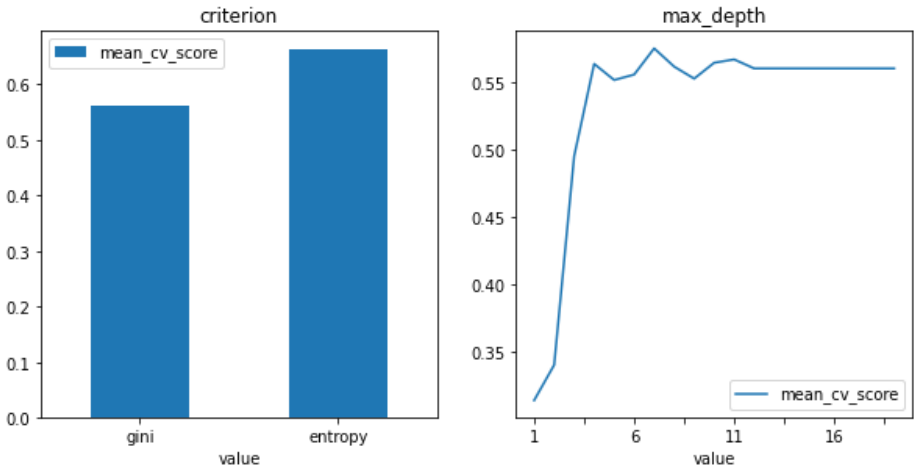
Задание а.

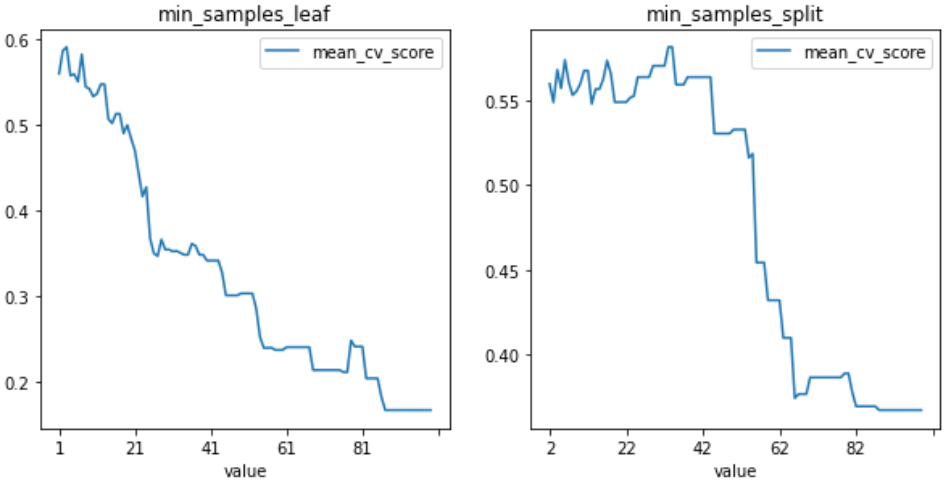
Дерево решения:



Это дерево имеет избыточное количество узлов, что указывает на наличие признаков переобучения.

Исследуем зависимость точности классификации от критерия расщепления, максимальной глубины, минимального разделения выборки и минимального количества samples:

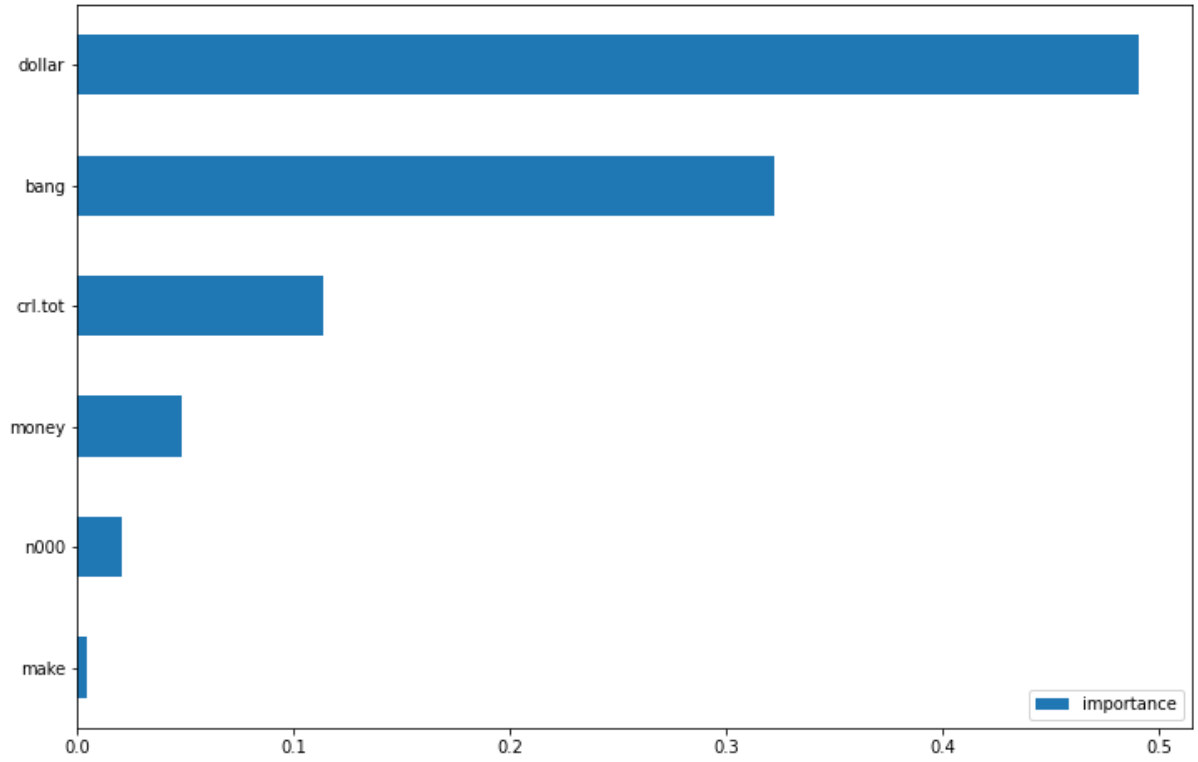




Задание b.

Дерево решения:

Определяем наиболее влияющие признаки:



Качество классификации:

```
Train accuracy:  0.9632779285104514
Test accuracy:  0.8254024807824492
```

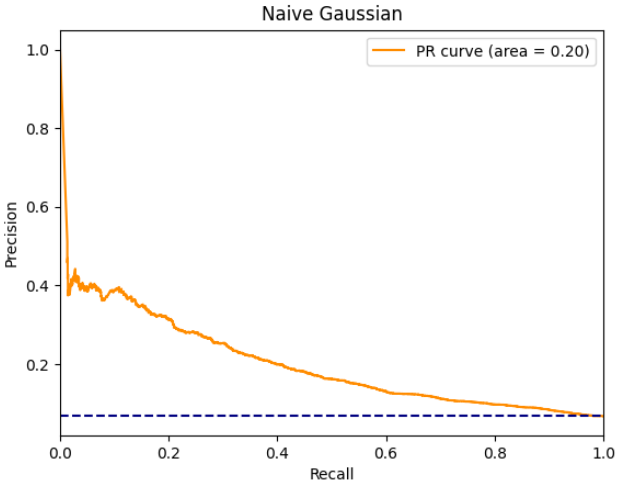
Задание 6.

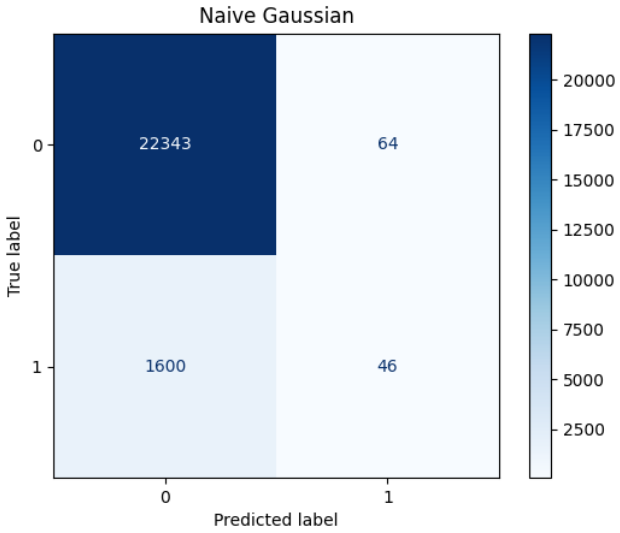
6. Загрузите набор данных из файла bank_scoring_train.csv. Это набор финансовых данных, характеризующий физических лиц. Целевым столбцом является «SeriousDlqin2yrs», означающий, ухудшится ли финансовая ситуация у клиента. Постройте систему по принятию решения о выдаче или невыдаче кредита физическому 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.

Подходящими метриками являются Precision (доля правильно предсказанных объектов среди всех предсказаний истинного класса) и Recall (доля правильно предсказанных объектов из объектов из предсказанных как принадлежащих классу). Построим PR-кривые для каждого классификатора.

Naive Gaussian.

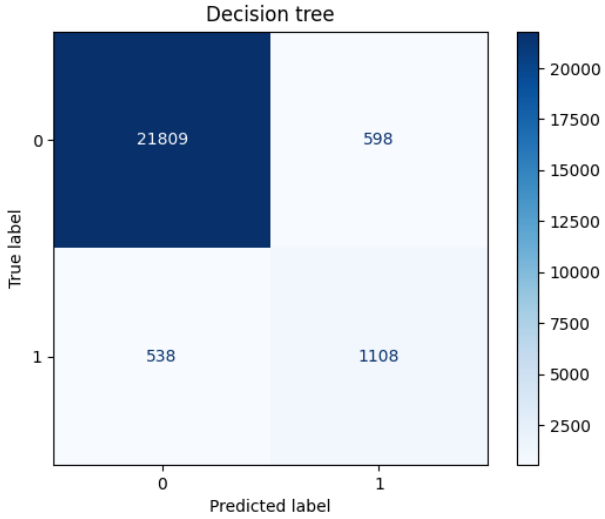
```
Naive Gaussian
Accuracy: 0.9308194404024446
Presicion: 0.41818181818181815
Recall: 0.027946537059538274
```

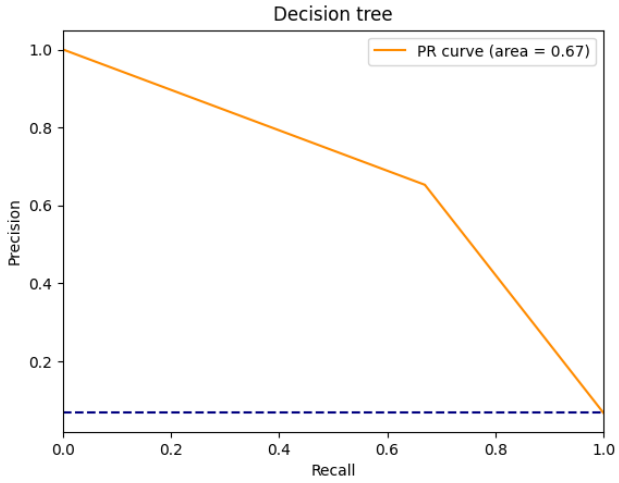




Decision tree.

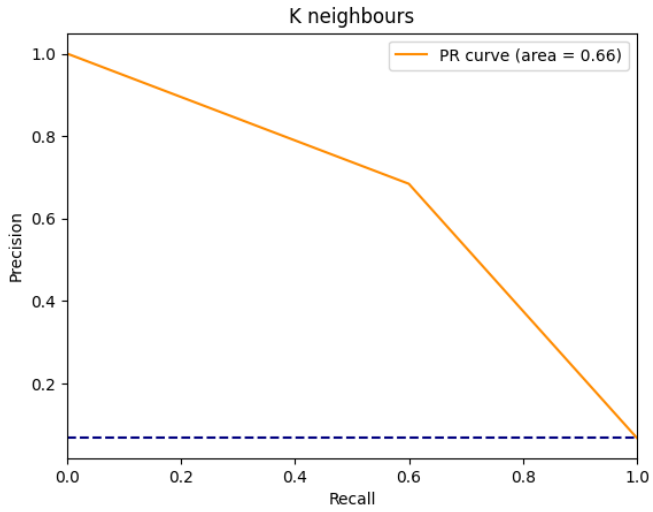
```
Decision tree
Accuracy: 0.9520226167214069
Presicion: 0.6418685121107266
Recall: 0.6761846901579587
```

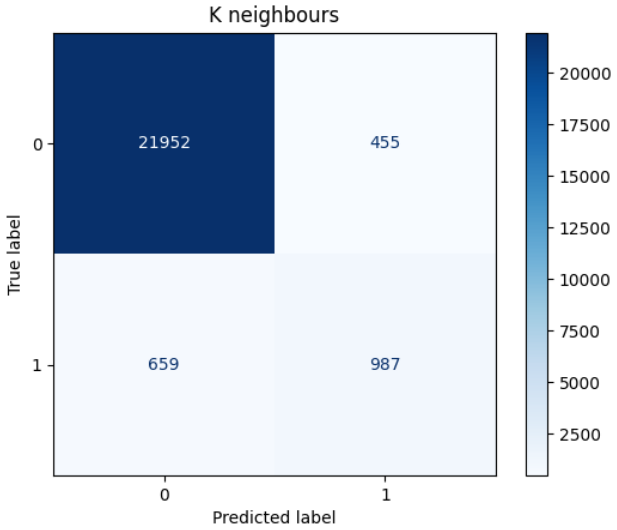




K neighbours.

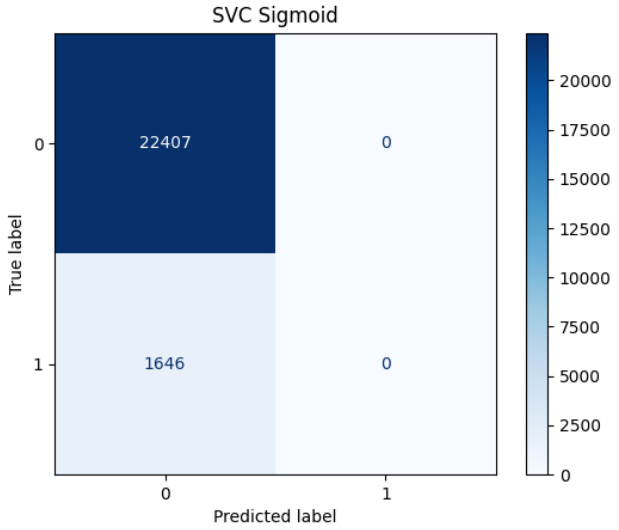
```
K neighbours
Accuracy: 0.9536856109425019
Presicion: 0.6844660194174758
Recall: 0.5996354799513973
```

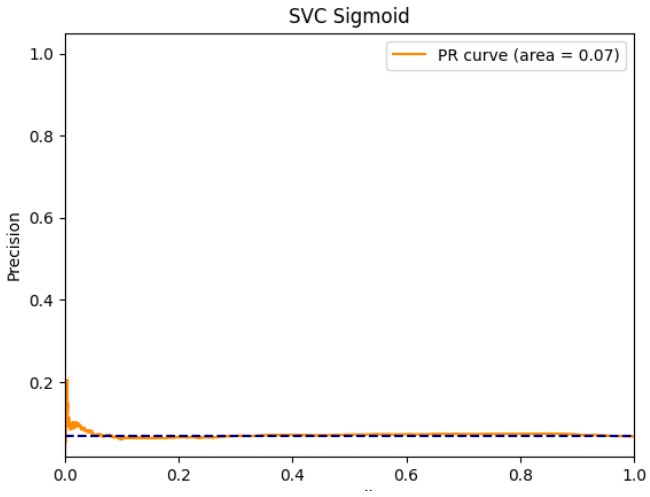




SVC Sigmoid.

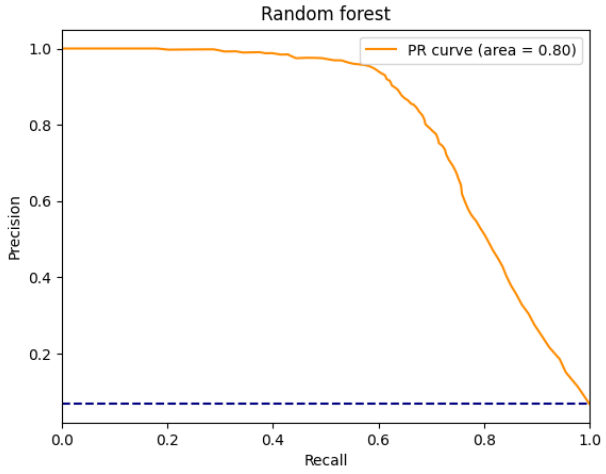
```
SVC Sigmoid
0.9315677878019374
Presicion: 0.0
Recall: 0.0
```

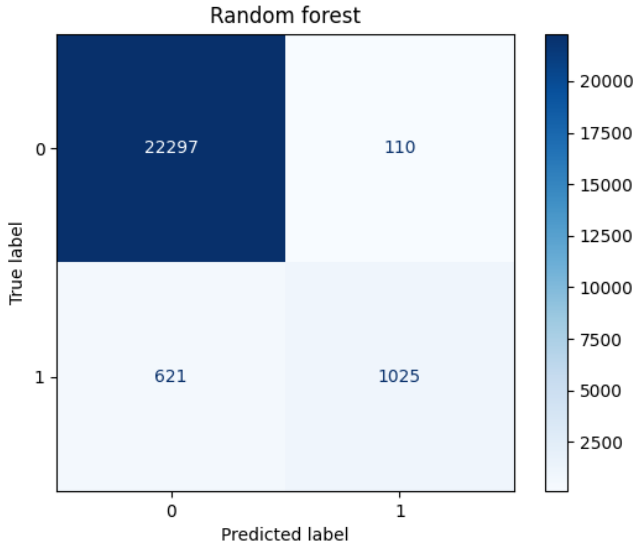




Random forest.

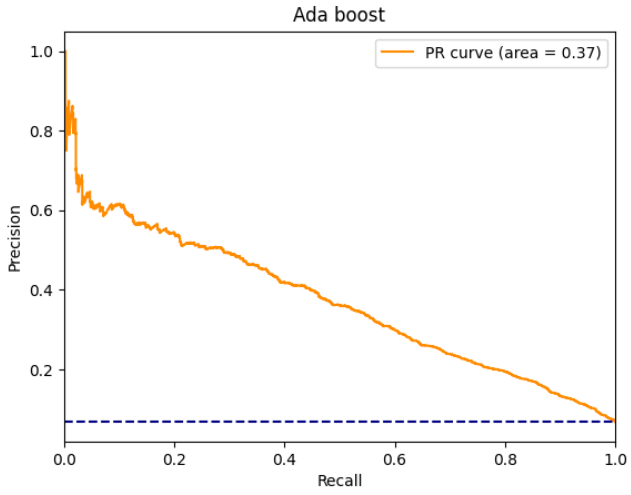
```
Random forest
Accuracy: 0.9700245291647611
Presicion: 0.9082082965578111
Recall: 0.6251518833535844
```

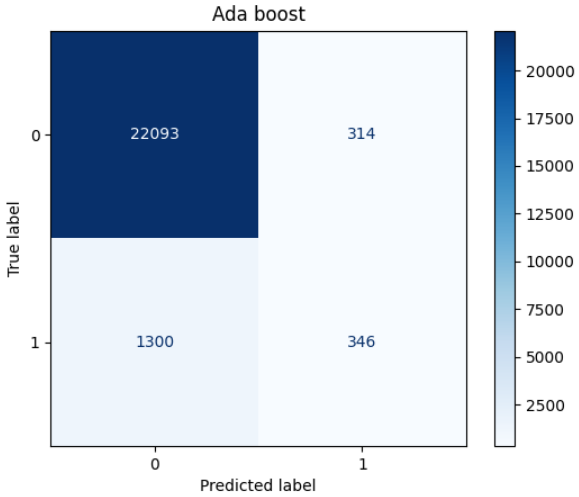




Ada boost.

```
Ada boost
Accuracy: 0.9328981831788135
Presicion: 0.5242424242424243
Recall: 0.21020656136087484
```





При анализе данного набора данных наилучший результат был достигнут с использованием метода Random Forest, и это было определено на основе метрики Precision.