Отчет о практическом задании Метрические алгоритмы классификации

Практикум 317 группы, ММП ВМК МГУ

Пшеничников Глеб Викторович

Москва
Октябрь 2024

Содержание

1 Введение

Данное задание посвящено исследованию метрических алгоритмов классификации и методов работы с изображениями. Исследование будет проводиться на наборе рукописных текстов *MNIST*[1]. Целью данной работы является сравнение алгоритмов поиска ближайших соседей, метрик, по которым считаются расстояния между объектами, и преобразований изображений из тренировочной и тестовой выборок для достижение лучшего качества предсказаний модели. В этом исследовании проведен анализ вышеперечисленных сущностей и для каждой их них определены оптимальные параметры по качеству и времени выполнения.

2 Пояснение к задаче

Пусть имеется выборка объектов:

$$\mathbb{X} = \{(x_1, y_1), ..., (x_n, y_n)\},\ x_i \in X,\ y_i \in Y\ \forall \in i = \overline{1, n}$$

Методом k-го ближайшего соседа называется алгоритм метрической классификации, в котором таргет объектов определяется через его k ближайших соседей. Метод справедлив, так как мы предполагаем, что справедлива гипотеза компактности - объекты с одинаковым таргетами находятся ближе по метрике, с разными - дальше друг от друга. То есть, упорядочив все объекты из тренировочной выборки по возрастанию расстояний до объекта, целевую переменную которой мы предсказываем, первые k из них будут ближайшими соседями и исходя из значений их целевой переменной делаем предсказание. Эта модель имеет вид:

$$a(x; \mathbb{X}) = \underset{y \in Y}{argmax} \sum_{i=1}^{n} I[y_i = y] w(i, x)$$

Решаемая задача заключается в построении модели классификации, выдающую наибольшую точность и работающую за наименьшее время. Для этого нужно оптимально задать параметры модели:

- Число ближайших соседей k
- Метрика, по которой вычисляются расстояния между объектами
- Вес i-го ближайшего соседа $w(i, x)$
- Алгоритм поиска ближайшего соседа

2.1 Число ближайших соседей k

Этот параметр задает количество ближайших объектов, таргет которых будет участвовать в определении класса объекта, для которого делается предсказание.

2.2 Метрика

В рамках исследования будут рассматриваться две метрики: евклидова и косинусная. По ней будет рассчитываться расстояние между объектами.

Расстояние по евклидовой метрике вычисляется по формуле:

$$q(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Расстояние по косинусной метрике вычисляется по формуле:

$$q(x, y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}}$$

2.3 Вес i-го ближайшего соседа

Вес определяет насколько силен голос соседа в определении таргета объекта. В рамках задачи рассматриваются два значения веса:

- $\alpha^i, \alpha \in 0, 1$
- $\frac{1}{\epsilon + q(x, x_i)}$, где $\epsilon = 10^{-5}$, а $q(x, x_i)$ - расстояние до i-го объекта из k ближайших соседей

2.4 Алгоритм поиска ближайшего соседа

В рамках исследования рассматриваться следующие алгоритмы поиска ближайшего соседа:

- **Balltree**(используется библиотека *sklearn*) - это древовидная структура. Суть алгоритма заключается в разбиение исходного пространства данных на вложенные гиперсферы. Этот подход позволяет более эффективно отсекать большие области пространства, в которых отсутствуют ближайшие соседи для точек. Для этого алгоритма можно использовать только евклидову метрику вычисления расстояний.
- **Kdtree**(используется библиотека *sklearn*) - это древовидная структура, в которой происходит разбиение исходного пространства данных на вложенные гиперплоскости. Этот поход так же, как и *Balltree*, позволяет эффективно отсекать большие области пространства, в которых отсутствуют ближайшие соседи для точек. Для этого алгоритма можно использовать только евклидову метрику вычисления расстояний.
- **Brute Force**(используется библиотека *sklearn*) - основан на полном переборе объектов для определения k ближайших соседей. Можно использовать обе метрики вычисления расстояний.
- **Собственная реализация** - алгоритм основан на вычислении расстояний между любыми двумя объектами, сортировке этих расстояний и выделение из них k наименьших. Можно использовать обе метрики вычисления расстояний.

2.5 Точность предсказаний

Качество модели будет проверяться при помощи метрики *accuracy*:

$$accuracy = \frac{\text{кол-во правильных предсказаний}}{\text{общее кол-во предсказаний}}$$

3 Эксперименты

В данном исследовании проведено 6 экспериментов, в каждом и которых выявлены оптимальные параметры, методы, алгоритмы, которые используются для построения модели классификации рукописных текстов.

3.1 Скорость алгоритмов поиска ближайших соседей

Было проведено сравнение скоростей алгоритмов поиска ближайших соседей: *Balltree, Kdtree, Brute Force, Собственная реализация*. Эксперимент проводились на разном количестве признаков: на 10, 20, 100 (из 784 имеющихся), производился поиск *5*-ти ближайших соседей. Результаты эксперименты представлены в таблице:

	Balltree	**Kdtree**	**Brute Force**	**Собственная реализация**
10 признаков	1.83 сек	0.91 сек	6.85 сек	46.46 сек
20 признаков	5.13 сек	1.63 сек	6.99 сек	51.33 сек
100 признаков	59.64 сек	58.56 сек	7.50 сек	75.21 сек

Таблица 1: Время работы алгоритмов по поиску ближайших соседей на 10, 20, 100 признаках

Из таблицы видно, что наименьшее время на небольшом наборе признаков (10 и 20) показывает алгоритм *Kdtree*. При увеличении числа признаков до 100 наименьшее время требуется алгоритму *Brute Force*.

Собственная реализация показывает наибольшее время на любом количестве признаков (при этом разница во времени относительно велика), это происходит из-за неэффективности реализации поиска ближайших соседей. *Brute Force* наиболее стабильная модель и незначительно меняет время при увеличении количества признаков, а *Kdtree* и *Balltree* чувствительны к изменению числа признаков, и время выполнения значительно увеличивается. Такие результаты можно объяснить вычислительной сложностью алгоритмов.

Kdtree и *Balltree* имеют сложность построения $O(n \log n)$, сложность поиска k соседей (в среднем) $O(\log n+k)$. На 10 и 20 признаках *Kdtree* немного быстрее работает в силу архитектуры (использует гиперплоскости, а не гиперсферы). При 100 признаках оба алгоритма значительно повышают время выполнения и сравниваются - это происходит из-за "проклятия размерности".

Brute Force имеет сложность построения $O(1)$, сложность поиска k соседей (в среднем) $O(nkd)$, где n - общее количество имеющихся в рассмотрении точек, d- количество признаков. При малых количествах признаков показывает не лучшее время в силу того, что проверяются все точки. При возрастании числа признаков алгоритм становится сравнимым по скорости, так как линейно зависит от числа признаков - этим объясняется стабильность данного алгоритмы.

Собственная реализация имеет сложность построения $O(1)$, сложность поиска k соседей (в среднем) $O(nd + n \log n + k)$, где n - общее количество имеющихся в рассмотрении точек, d - количество признаков ($O(nd)$ для вычисления всех расстояний, $O(n \log n)$ для сортировки, $O(k)$ для выбора k наименьших расстояний). Этим объясняется длительность выполнения этого алгоритма на любом количестве признаков (имеет самую высокую сложность среди других представленных алгоритмов).

Итого, лучшим алгоритмом на небольшом количестве признаков (до 20) является ***Kdtree***, при большом количестве признаков (более 20) наименьшее время показывает модель ***Brute Force***.

3.2 Зависимость точности и скорости модели от k (количества ближайших соседей) и от метрики вычисления расстояний

В данном эксперименте сравнивались точность и время работы модели в зависимости от k(количества ближайших соседей) и от метрики вычисления расстояний(евклидова и косинусная). Для проведения исследования использовался алгоритм *Brute Force*, так как используется

все признаки (784 признака), и исходя из результатов эксперимента 3.1 (также этот алгоритм допускает возможность использовать обе метрики вычисления расстояний). Исследование проводилось при помощи кросс-валидации с 3 фолдами. Также модель будет использовать метод без весов для голосования ближайших соседей.

3.2.1 Зависимость точности от количества соседей

Для каждого k при помощи кросс-валидации вычислим точность предсказаний модели, как среднее значение точности на каждом фолде:

$$accuracy = \frac{accuracy_1 + accuracy_2 + accuracy_3}{3}$$

где $accuracy_i$ - точность предсказаний при обучении на всех фолдах кроме *i-го* и тестировании на *i-ом* фолде.

Результаты эксперимента представлены на графике 1

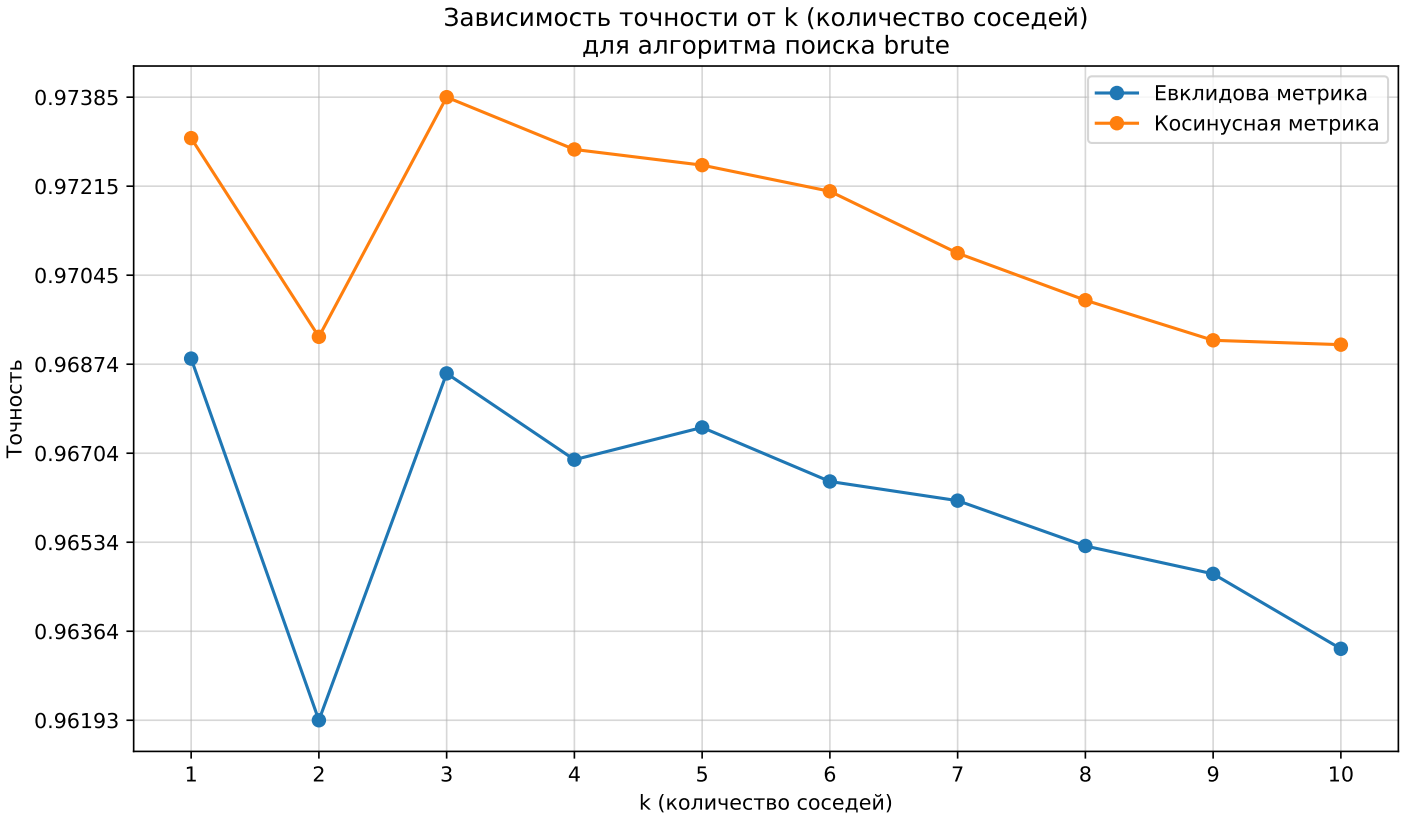


График 1: Зависимость точности модели от количества ближайших соседей и от метрики

Видно, что точность на максимальна при $k = 3$, далее с увеличением k точность снижается. Так происходит, потому при больших k модель учитывает большее количество шумов или может взять в качестве соседа отдаленный объект. То есть большое k может переобучить модель и размыть границы одного класса. При значениях k 1 и 2 точность ниже, так как имеют место шумы и случайные флуктуации.

Так же при k равному 2 мы видим резкое понижение точности, это происходит по нескольким причинам: в силу честности имеются случаи, когда модель имеет 1 голос за один класс и 1 голос за другой - следовательно, модель не знает, какой класс выбрать, появляется случайность в выборе класса, что снижает точность предсказаний; в силу малого значения k модель

неустойчива к выбросам; на границе класса решение еще более неустойчиво, так как малое изменение признака может повлечь неправильное предсказание; два соседа не могут дать полную и точную информацию о классе.

Резкий скачок вверх наблюдается при k равного 3, это можно объяснить тем, что это является относительно граничным случаем между сильной неустойчивостью модели (при k равного 1 и 2) и размытием границ классов (при больших k).

3.2.2 Зависимость точности от метрики

Зависимость точности от метрики показаны на графике 1. Косинусная метрика показывает более высокую точность на всем диапазоне k. Так происходит по следующим причинам:

- Косинусная метрика учитывает соотношение значение, а не их абсолютное значение. Это может быть полезно, потому что одну и ту цифру можно написать с разной яркостью, силой нажатия, контрастностью чернил - что приводит к тому, что значения признаков увеличиваются или уменьшаются, что может привести модель в заблуждение. Косинусная метрика позволяет модели учитывать относительные величины, что является нормализацией признаков.

- В признаках много нулевых значений, косинусная метрика фокусируется на ненулевых значениях (так как слагаемое в скалярном произведении, стоящее в числители формулы косинусной метрики, соответствующее нулевому признаку, равно нулю)

- Косинусная метрика хорошо масштабируема, так как учитывает угол между векторами, а не их длину

3.2.3 Зависимость времени выполнения от метрики

Результаты эксперимента представлены в виде Графика 2.

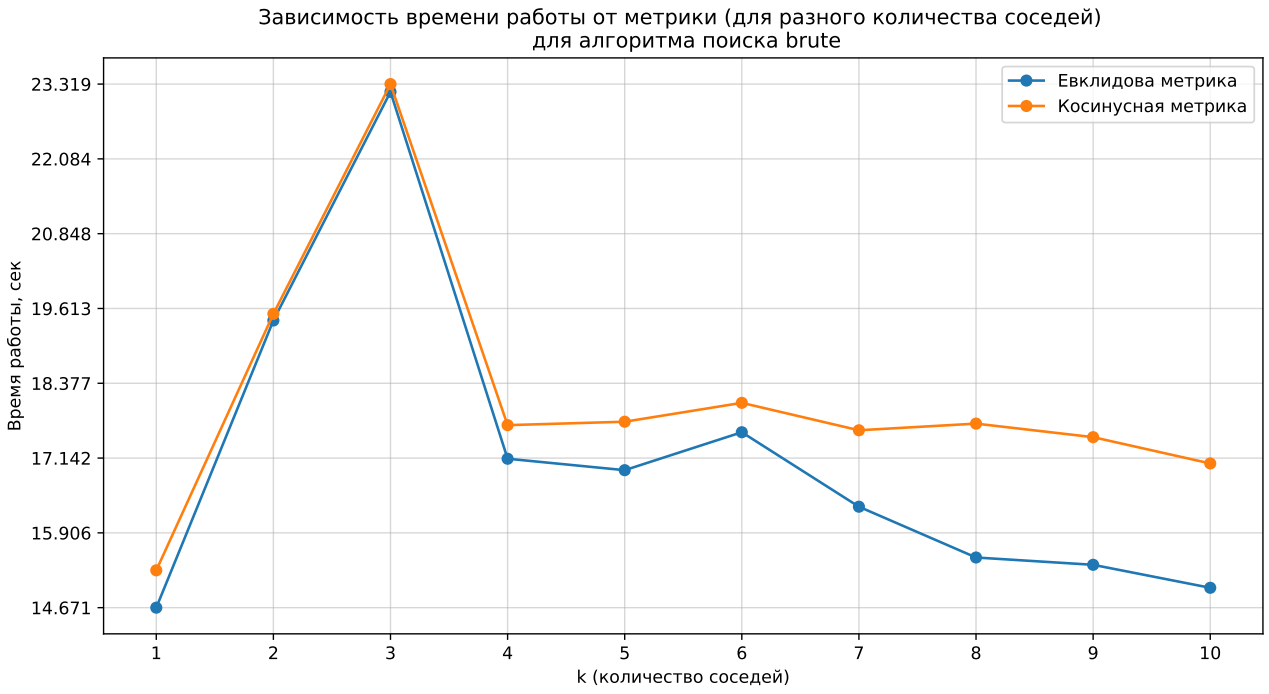


График 2: Зависимость времени модели от количества ближайших соседей и от метрики

Видно, что косинусная метрика работает немного медленнее, особенно это видно при увеличении k, когда вычислений становится больше. Это происходит потому, что при вычислении косинусной метрики выполняется больше арифметических операций(вычисление скалярных произведений, норм векторов, деление и вычитание), в евклидовой норме их меньше(возведение в квадрат, вычитание, извлечение корня). Так же в косинусной метрики происходит работа с числами с плавающей точкой (из-за деления), что так же может замедлить вычисление.

Итого, можно сделать вывод, оптимальным значением (по качеству) k для модели (с методом голосования без весов) является значение 3, а оптимальной метрикой является косинусная метрика. По времени косинусная метрика незначительна медленнее, чем евклидова.

3.3 Сравнение взвешенного метода голосования и метода без весов

В этом эксперименте требуется выявить лучший метод голосования (с использованием весов $\frac{1}{\epsilon+q(x,x_i)}$ или без них) среди k ближайших соседей. Для эксперимента использовался алгоритм поиска ближайших соседей *Brute Force*. Вычислялись точность предсказаний и время работы модели, была использована кросс-валидацию с 3 фолдами. Итоговая точность и время работы модели при конкретных параметрах вычислялась как среднее соответствующих величин, полученных на фолдах.

3.3.1 Сравнение точности взвешенного метода голосования и метода без весов

Результаты представлены в виде графика:

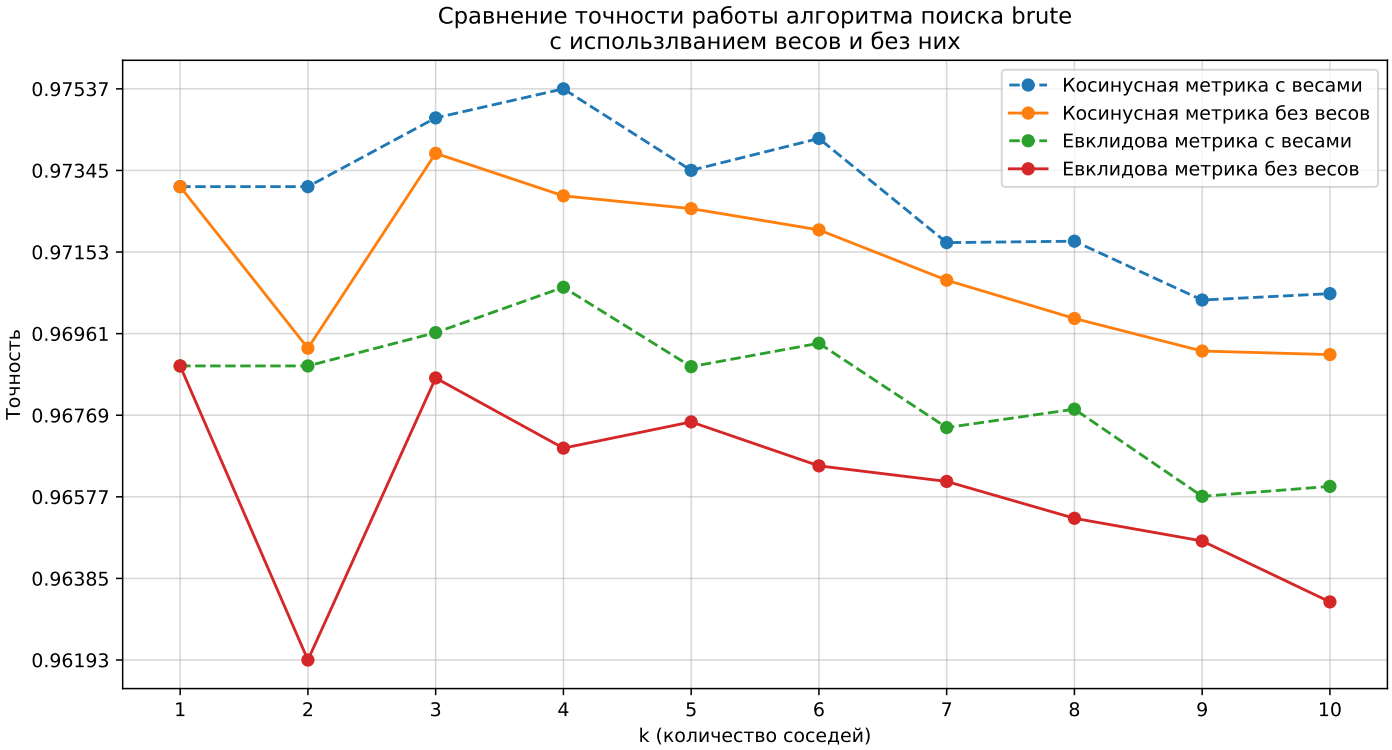


График 3: Точность взвешенного метода голосования и метода без весов для разных метрик, k

Исходя из полученных результатов, можно сделать следующие выводы:

- Добавление весов улучшает точность модели для всего диапазона k и для обоих метрик вычисления расстояний. Этот результат говорит о том, что из всех признаков можно выделить более информативные (множество пикселей)
- Использование косинусная метрика приводит к большей точности (подробно проанализировано в 3.2.2)
- При k более 4 точность падает, как и при k равном 1 и 2 (подробно проанализировано в 3.2.1)
- Так же стоит заметить, что лучшее качество дает модель при k равным 4, косинусной метрикой с использованием весов. Для дальнейших исследований будет использовать именно эти параметры.

3.3.2 Сравнение времени выполнения взвешенного метода голосования и метода без весов

Результаты представлены в графика:

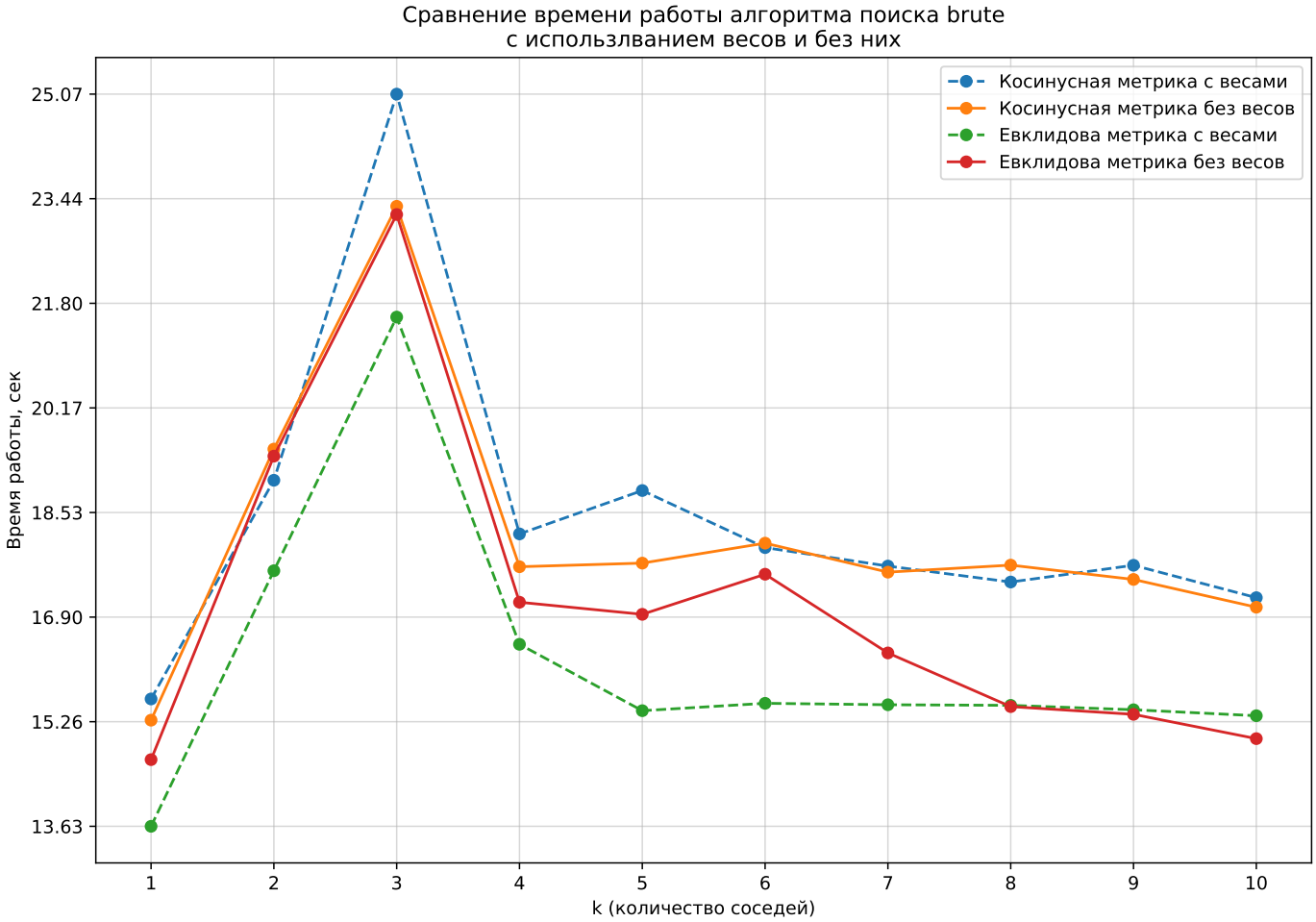


График 4: Время выполнения взвешенного метода голосования и метода без весов для разных метрик, k

Из данных результатов можно сказать, что все самым быстрым является метод с весами и евклидовой метрикой (почти на всем диапазоне k показывает наименьшее время). Косинусная метрика с весами требует больше всего времени (хотя разница незначительна). Так же заметно, что все методы находятся друг от друга в диапазоне 3-5 секунд, это объясняется стабильностью модели *Brute Force* (подробнее 3.1).

Итого, можно сделать вывод, что самыми оптимальными параметрами для модели будет: косинусная метрика при k равным 4 с использованием взвешенным методом голосования.

3.4 Тестирование модели с лучшими параметрами, анализ ошибок

Цель этого эксперимента протестировать модель с лучшими параметрами на тестовых данных6, провести кросс-валидацию, сравнить с лучшей моделью из Интернета, проанализировать ошибки. Лучшей моделью, как было выяснено в предыдущем эксперименте является *Brute Force* с косинусной метрикой при k равным 4 с использованием взвешенным методом голосования.

Имеем следующие результаты:

	accuracy
Тестовая выборка	0.9781
Тренировочная выборка	1.0
Кросс-валидация фолд 1	0.9756
Кросс-валидация фолд 2	0.97595
Кросс-валидация фолд 3	0.97455
Кросс-валидация среднее значение	0.9754
Лучшая модель из Интернета	0.9982

Таблица 2: Точность модели с лучшими параметрами

Можно сделать вывод, что на тренировочном датасете произошло переобучение (так как точность 1). Точность на каждом фолде, средняя точность в кросс-валидации, точность на тестовой выборке примерно одинакова, значит модель адаптируется к новым данным. Но при этом полученный результат далек от лучшей реализации - лучший результат (взятый из Интернета) больше на 0.2.

Для разбора ошибок, проанализируем матрицу ошибок - график 5. Можно сделать выводы:

- Наиболее часто модель ошибается, когда анализирует 0, 1, 8, 9 (в силу схожести написания)
- Наиболее частые ошибки: при анализе 9 предсказывает 4 и 7; при анализе 8 предсказывает 3; при анализе 5 предсказывает 8
- В целом ошибки распределены равномерно (за исключением ошибок из предыдущего пункта) - от части это может происходить из-за наличия шумов и выбросов
- Модель хорошо предсказывает 0, 2, 4 (в силу того, что остальные цифры при даже неаккуратном написании не будут похожи на них)

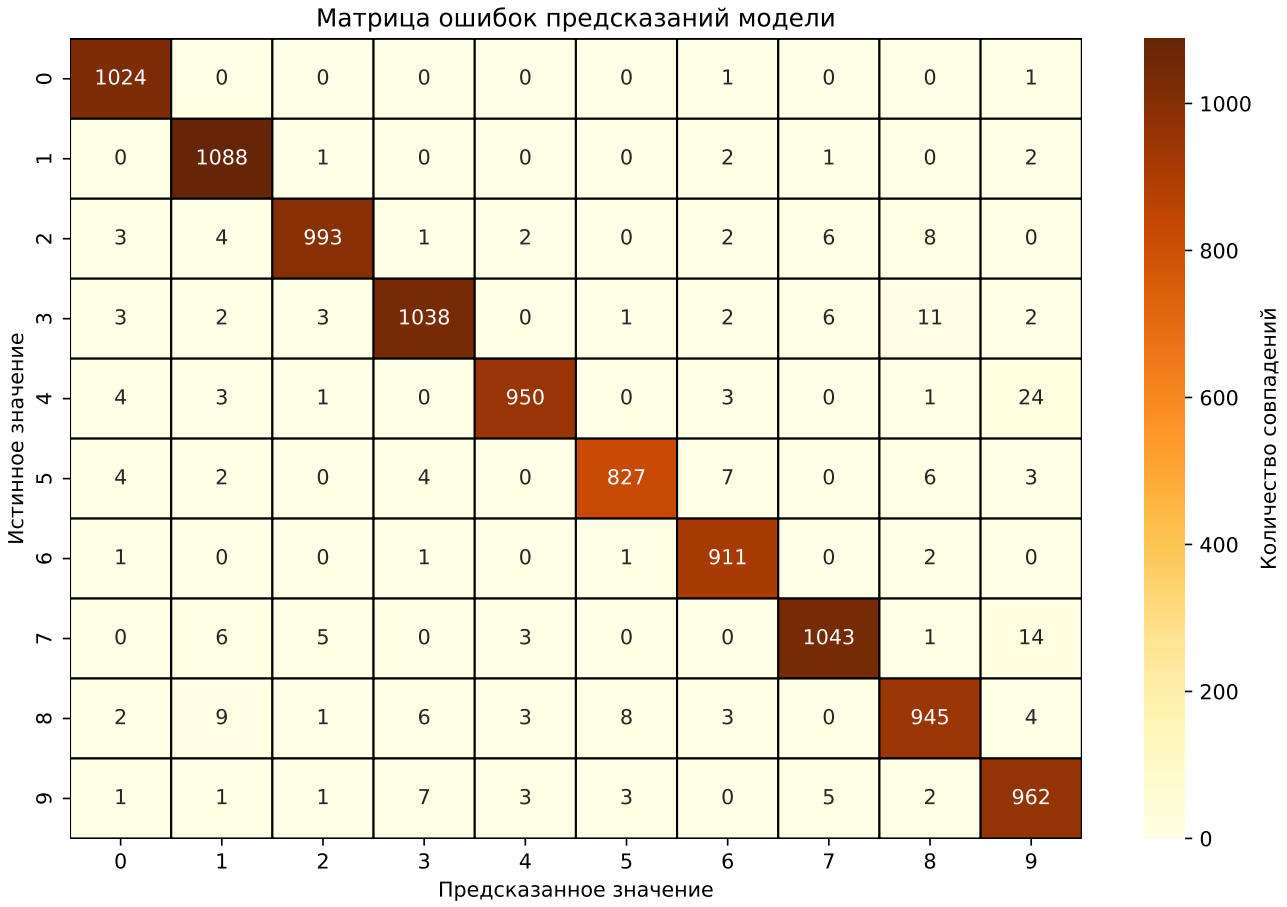
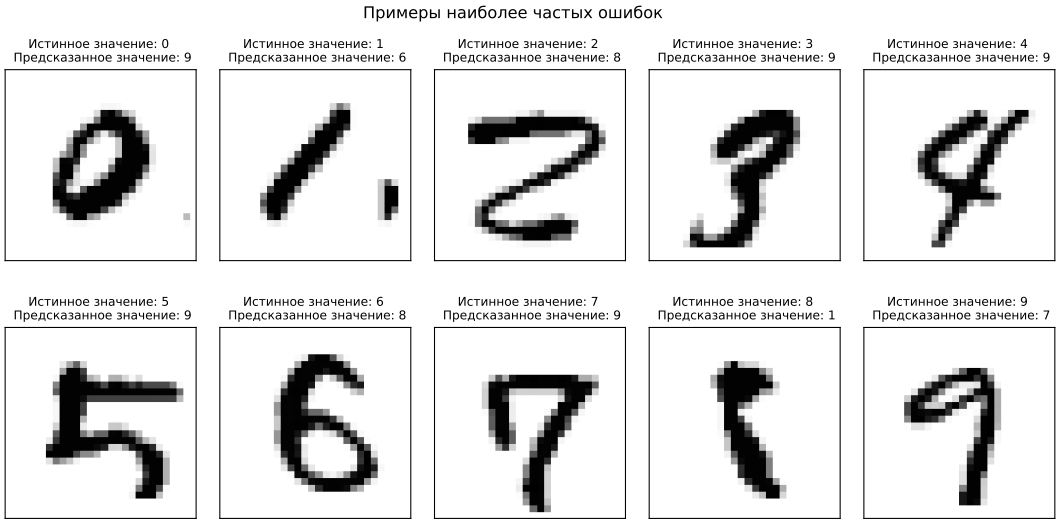


График 5: Матрица ошибок

Рассмотрим примеры наиболее частых ошибок для каждой из цифр:



Можно сделать вывод, что наиболее частые ошибки возникают из-за небрежного написания (например 8, 9), шумов (например 1) или из-за того, что цифра вылезает за границу (например 3, 5).

Итого, из этого эксперимента можно сделать вывод, что ошибки возникают из-за встречающийся некорректности данных. Следовательно, для улучшения качества можно попробовать преобразовать данные.

3.5 Аугментация тренировочной выборки

В данном эксперименте проведен анализ, как добавление преобразований изображений в тренировочную выборку влияет на точность модели. Будут рассмотрены следующие преобразования:

- Поворот на 5, 10, 15 (в каждую из двух сторон)
- Смещение на 1, 2, 3 пикселя (по каждой из двух размерностей)
- Дисперсия фильтра Гаусса: 0.5, 1, 1.5
- Морфологические операции: эрозия, дилатация, открытие, закрытие с ядром 2

Для поворотов, смещения и дисперсии фильтра Гаусса применена кросс-валидация с 3 фолдами для определения лучшего параметра. В качестве точности для каждого параметра бралось среднее значение точности на фолдах.

3.5.1 Повороты

Результаты точности на кросс-валидации занесены в таблицу 3

Величина поворота	**accuracy**
5	0.85
10	0.83
20	0.81

Таблица 3: Точность модели при поворотах тренировочной выборки

Исходя из результатов можно сказать, что поворот на 5 (сторона поворота определялась случайным образом) имеет лучшую точность. Преобразовав тренировочную выборку при помощи поворота на 5, получаем точность на тестовой выборке равной: **0.9781**

3.5.2 Смещение

Результаты точности на кросс-валидации занесены в таблицу 4

Величина смещения	**accuracy**
1	0.9725
2	0.9680
3	0.9676

Таблица 4: Точность модели при смещениях тренировочной выборки

При смещении на 1 по обоим осям(направление смещения определялась случайным образом) имеет лучшую точность. Преобразовав тренировочную выборку при помощи смещения на 1, получаем точность на тестовой выборке равной: **0.9803**

3.5.3 Дисперсии фильтра Гаусса

Размер ядра фильтра Гаусса подберем визуально (преобразование не должно сильно портить объекты). Рассмотрим преобразования с разными ядрами для каждой *sigmaX* (0.5, 1, 1.5).

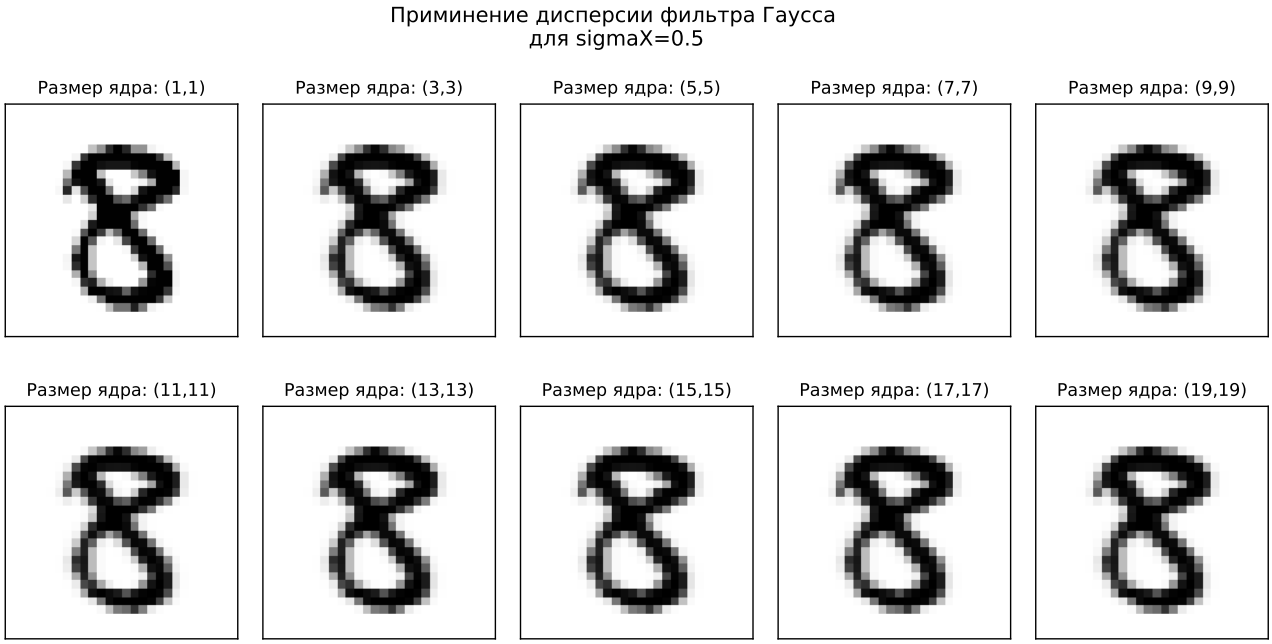


График 6: Дисперсии фильтра Гаусса для sigmaX = 0.5

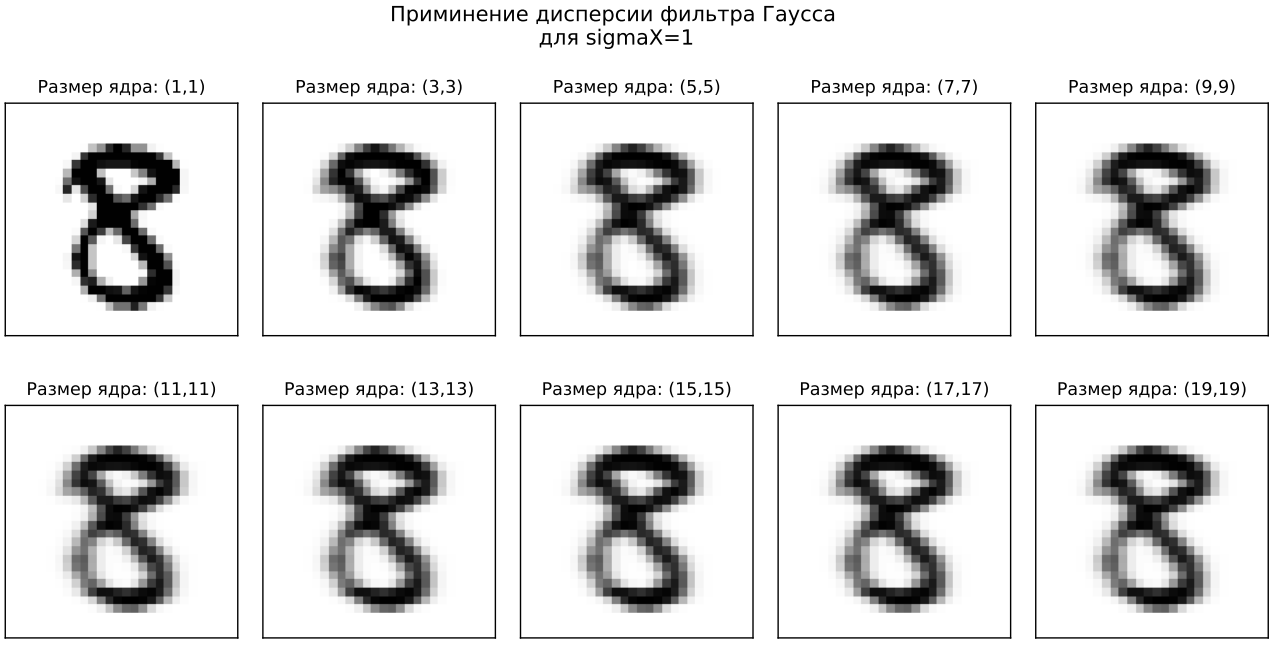


График 7: Дисперсии фильтра Гаусса для sigmaX = 1

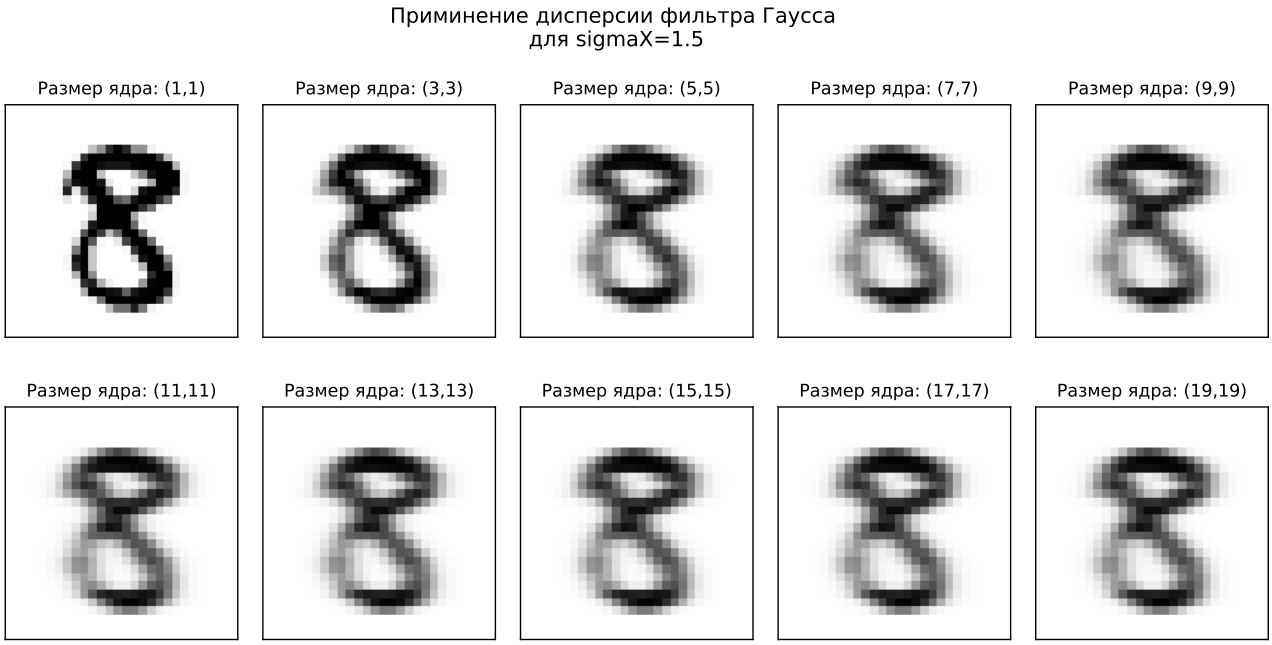


График 8: Дисперсии фильтра Гаусса для sigmaX = 1.5

Анализируя результаты, можно сделать вывод, что для sigmaX = 0.5 оптимально использовать ядро размера 7; для sigmaX = 1 - ядро размера 9; для sigmaX = 1.5 - ядро размера 11. Применим кросс-валидацию для данных значений. Результаты занесены в таблицу 5

Величина sigmaX	**accuracy**
0.5	0.9918
1.0	0.9913
1.5	0.9863

Таблица 5: Точность модели при применении дисперсии фильтра Гаусса к тренировочной выборке

При применении дисперсии фильтра Гаусса с sigmaX = 0.5 имеем лучшую точность. Преобразовав тренировочную выборку при помощи дисперсии фильтра Гаусса с sigmaX = 0.5 и ядром размера 7 , получаем точность на тестовой выборке равной: **0.9764**

3.5.4 Морфологические признаки

Преобразуем тренировочную выборку, используя каждый их рассматриваемых морфологических преобразований (с ядром 2). Результаты занесены в таблицу 6:

Преобразование	**accuracy**
Эрозия	0.9779
Дилатация	0.9807
Открытие	0.9796
Закрытие	0.9795

Таблица 6: Точность модели при применении морфологических преобразований к тренировочной выборке

3.5.5 Комбинация преобразований

Для определения оптимальных комбинаций, построим матрицу ошибок для каждого преобразования 10

Исходя из полученных результатов, можно предположить что оптимальными комбинациями будут: смещение + открытие, дилатация + смещение, дилатация + смещение + открытие, дилатация + дисперсия + смещение. Эти комбинации были составлены исходя из предположения, что, если совместить преобразования, каждое из которых дополняет другие (то есть не ошибаются на тех цифрах, на которых ошибаются другие преобразования), то получим более высокую точность. Для этих комбинаций точность 7 и матрица ошибок будут следующими 9:

Комбинации	**accuracy**
смещение + открытие	0.981
дилатация + смещение	0.9809
дилатация + смещение + открытие	**0.9819**
дилатация + дисперсия + смещение	0.9809

Таблица 7: Точность модели при комбинации преобразований

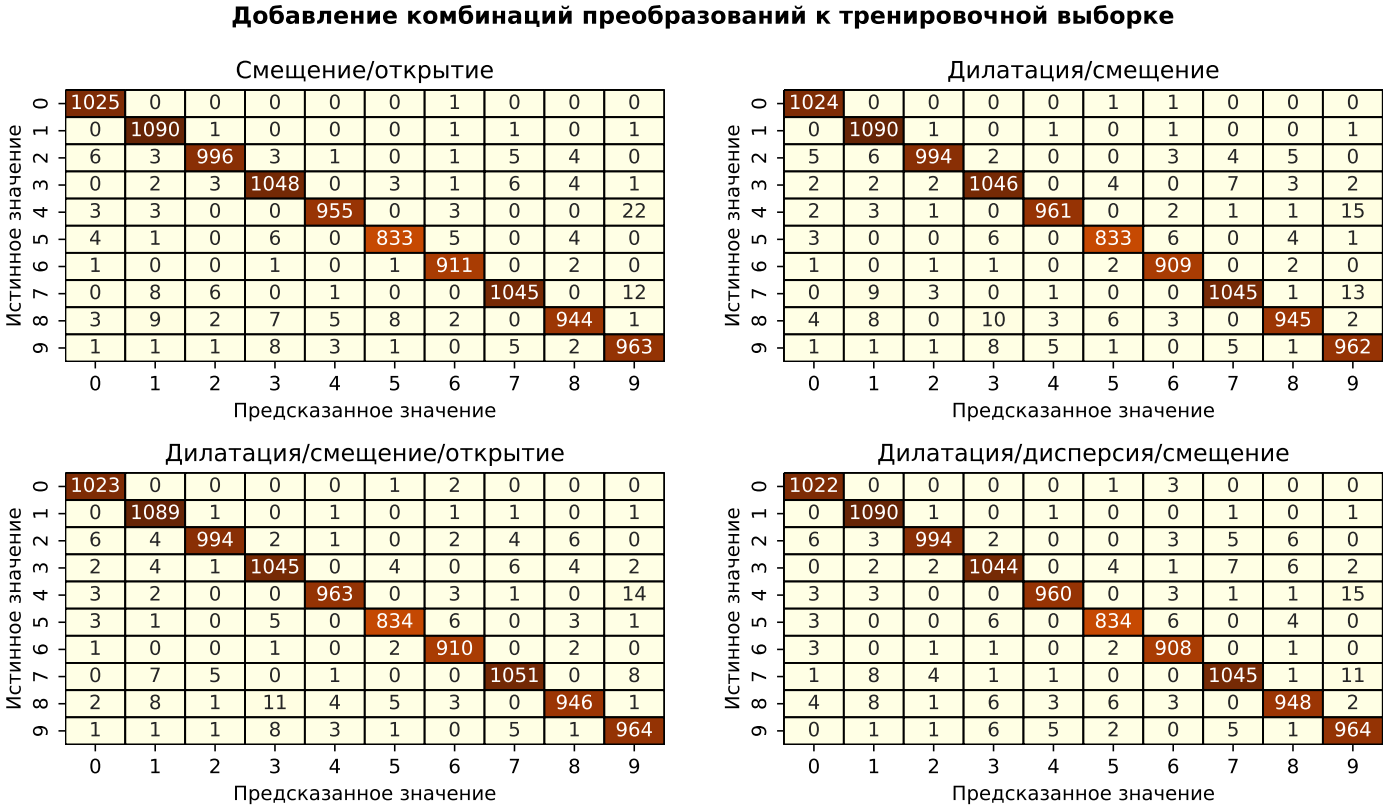


График 9: Матрица ошибок для комбинаций преобразований

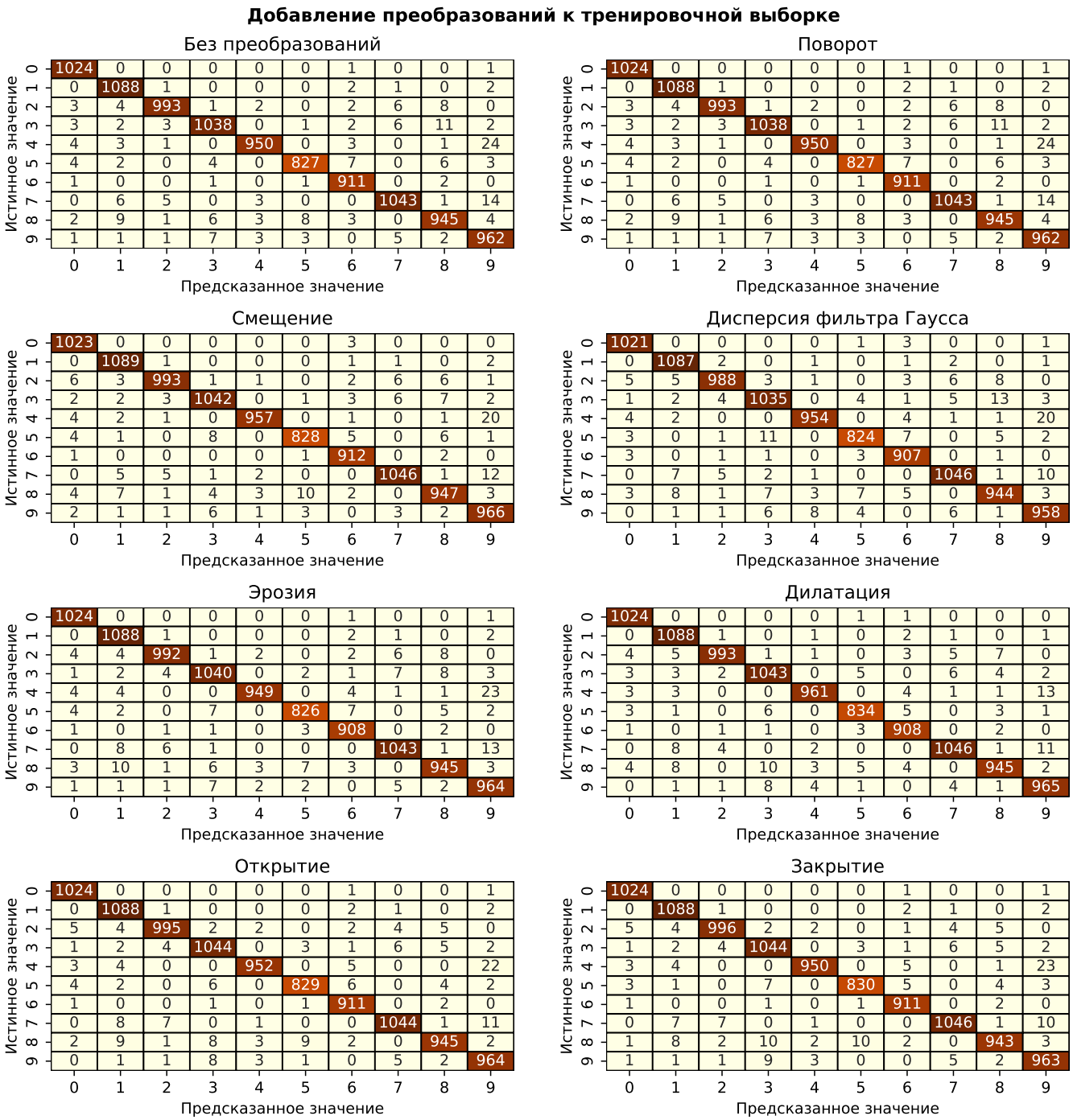


График 10: Матрица ошибок для всех преобразований тренировочной выборки

Из полученных значений видно, что точность действительно повысилась при комбинациях, а ошибки стали реже (то есть каждое преобразование отвечает за свою ошибку и убирает ее). Итого, общая таблица со всеми проведенными в данной эксперименте преобразованиями и их комбинациями 8

Преобразования	**accuracy**
Поворот на 5	0.9781
Смещение на 1	0.9803
Дисперсия фильтра Гаусса	0.9764
Дилатация	0.9807
Открытие	0.9796
Закрытие	0.9795
смещение + открытие	0.981
дилатация + смещение	0.9809
дилатация + смещение + открытие	**0.9819**
дилатация + дисперсия + смещение	0.9809

Таблица 8: Точность модели с преобразованиями

Вывод: для достижения наибольшей точности тренировочную выборку следует расширить при помощи комбинации преобразований: **дилатация + смещение + открытие**, полученная точность **0.9819**

3.6 Преобразования тестовой выборки

Цель эксперимента будут ли разумны преобразования изображений из тестовой выборки для улучшения точности предсказаний.

В данном эксперименте применим все преобразования из предыдущего эксперимента к тестовой выборке. Для каждой преобразованной выборки вычислим предсказания модели. Затем путем голосования (выбором наиболее встречающего класса в предсказаниях для конкретного теста) из полученных предсказаний сформируем итоговый вердикт. Точность при таком способе: **0.9748** Попробуем выбрать из преобразований некоторые признаки, которые путем описанного выше алгоритма дадут наилучшую точность. Для этого переберем все возможные комбинации преобразований, посчитаем для них accuracy и выберем три наилучшие комбинации. При следующих комбинациях точность наибольшая:

- **НАБОР 1**: смещение на 1 + смещение на 2 + смещение на 3 + дисперсия фильтра Гаусса с sigmaX=0.5 + дисперсия фильтра Гаусса с sigmaX=1.0 + дилатация
- **НАБОР 2**: смещение на 2 + смещение на 3 + дисперсия фильтра Гаусса с sigmaX=0.5 + дисперсия фильтра Гаусса с sigmaX=1.0 + ерозия
- **НАБОР 3**: смещение на 1 + смещение на 2 + смещение на 3 + дисперсия фильтра Гаусса с sigmaX=0.5 + дисперсия фильтра Гаусса с sigmaX=1.0 + эрозия + дилатация

Точность при этих комбинациях будет следующей:

Преобразования	**accuracy**
Набор 1	0.9775
Набор 2	0.9780
Набор 3	**0.9781**

Таблица 9: Точность модели при комбинации преобразований для тестовой выборки

Лучшая точность при данном методе была достигнута на наборе 2: **0.9781**, но по сравнению точностью на оригинальной тестовой выборке (без преобразований) 0.9781, этот метод не дал улучшений точности.

4 Выводы

В результате эксперимента была построена модель классификации рукописных текстов. Лучшая точность была достигнута после следующих параметрах модели/преобразованиях:

- Метрика вычисления расстояний: **косинусная**
- Использование взвешенного голосования при определении класса (вес равен $\frac{1}{\epsilon+q(x,x_i)}$, где $\epsilon = 10^{-5}$, а $q(x, x_i)$ - расстояние до i-го объекта из k ближайших соседей)
- k равное 4
- Добавление в тренировочную выборку преобразованных изображений при помощи: дилатация + смещение + открытие

Лучшее полученная точность *accuracy*:**0.9819**. Так же стоит отметить, что без добавления в тренировочную выборку преобразований, точность была **0.9781** - тоже достаточно высокая точность. А метод преобразования тестовой выборки не дал желаемого результата.

5 Заключению

В рамках данного задания было проведено знакомство с основными инструментами создания модели, основанной на метрических алгоритмах классификации.

Была построена модель классификации рукописных текстов *mnist*. Написаны метрики, кросс-валидация и функция, создающая фолды. Был проведен анализ времени выполнения и точности предсказаний модели с различными параметрами, с разными количеством признаков, с разными алгоритмами поиска ближайшего соседа. В каждом из 6 проведенных экспериментов был выявлен оптимальный параметр для модели. Была проведена работа по преобразованию изображений, изменению тренировочной, тестовой выборки для достижения наибольшей точности.

Список литературы

[1] MNIST: https://yann.lecun.com/exdb/mnist/