Отчет о практическом задании «Метрические алгоритмы классификации».

Практикум 317 группы, ММП ВМК МГУ.

Булкин Антон Павлович.

Октябрь 2024.

Содержание

1 Вступление

Данное практическое задание посвящено ознакомлению с метрическими методами машинного обучения на примере метода k ближайших соседей в задаче распознавания рукописных цифр из датасета MNIST.

Задачами задания являлись:

- Реализовать классификатор KNN на языке Python без использования библиотек *scipy* и *scikit-learn*
- Реализовать алгоритм поиска k ближайших соседей 4 методами: *kd_tree*, *ball_tree*, *brute* и *my_own*, состоящий в подсчете расстояний между объектами и поиске первых k порядковых статистик этих расстояний
- Реализовать кросс-валидацию и функцию, возвращающую скоринг по её итогам
- Провести 6 экспериментов на датасете изображений MNIST с тестовой выборкой размером 60 тыс. изображений, каждое из которых состоит из 28*28 пикселей, и тестовой выборкой из 10 тыс. изображений

2 Эксперименты

2.1 Исследование скоростей работы алгоритмов поиска ближайших соседей

2.1.1 Постановка задачи

Исследовать, какой алгоритм поиска ближайших соседей будет быстрее работать в различных ситуациях и почему.

Измерить для каждого алгоритма поиска (*kd_tree*, *ball_tree*, *brute* и *my_own*) время нахождения 5 ближайших соседей для каждого объекта тестовой выборки по евклидовой метрике. Выбрать подмножество признаков, по которому будет считаться расстояние, размера 10, 20, 100 (подмножество признаков выбирается один раз для всех объектов, случайно). Проверить все алгоритмы поиска ближайших соседей, указанные в спецификации к заданию.

2.1.2 Реализация

Проанализировано время работы всех перечисленных выше алгоритмов в зависимости от размера признакового пространства(10, 20, 100). (См. рис. 1) Признаки выбирались случайным образом из признакового пространства оригинального дастасета MNIST размера 784 при помощи встроенной функции библиотеки Numpy - *np.random.choice()*.

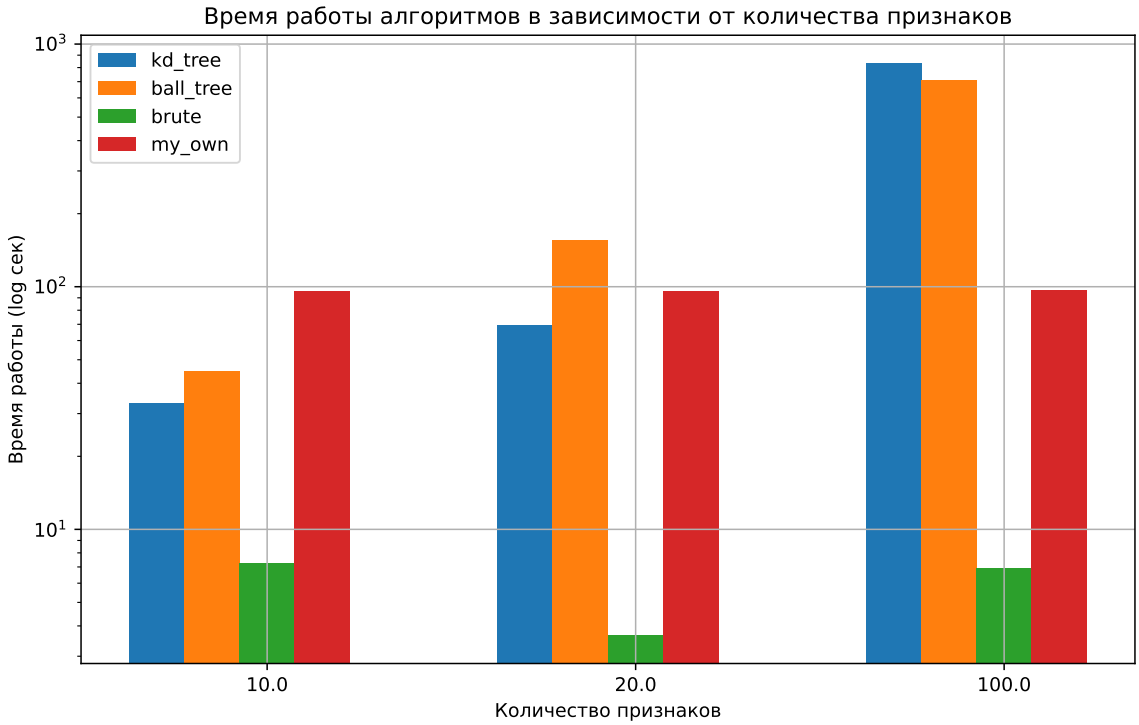

Рис. 1: Время работы алгоритмов поиска ближайших соседей в зависимости от размера признакового пространства

2.1.3 Выводы

Как можно заметить из графика:

- Алгоритм *brute* работает значительно быстрее других на любых размерах признаковых пространств.

- Алгоритмы *kd_ tree* и *ball_ tree* работают быстрее алгоритма *my_ own* на признаковых пространствах небольшой размерности: 20 и 50. Однако, уступают другим реализациям на большем количестве признаков из-за проклятия размерности, при котором расстояния между точками становятся почти неразличимыми, что усложняет работу данных алгоритмов, т.к. они используют оси координат/гиперсферы для создание деревьев, которые при больших размерностях становятся менее информативными.

- Алгоритм *brute* требует на выполнение программы не более 10 секунд на заданных размерностях признаковых пространств. Однако, на небольших объемах данных (использование только 10 признаков для поиска соседей) время его выполнения увеличивается. Возможно, из-за подсчета всех расстояний (логарифмическая сложность - O(n * d)). В то время такие алгоритмы как *kd_ tree* и *ball_ tree* разделяют пространство на подпространства, оптимизируя свою работу.

Исследование показало, что алгоритм *brute* работает быстрее всего на больших объемах данных. Таким образом, в дальнейших экспериментах будет использован данный алгоритм из-за его скорости выполнения.

2.2 Исследование точности алгоритма по кросс-валидации в зависимости от метрики и параметра k

2.2.1 Постановка задачи

Оценить по кросс-валидации с 3 фолдами точность (долю правильно предсказанных ответов) и время работы k ближайших соседей в зависимости от следующих факторов:

1. k от 1 до 10 (только влияние на точность).

2. Используется евклидова или косинусная метрика.

Дать ответы на следующие вопросы:

1. Какая метрика лучше себя показала в экспериментах и почему?

2. Есть ли на графике зависимости точности от количество соседей выбросы, резкие падения/повышения качества для одного значения k по сравнению с соседними? Если да, предположить причину появления этих выбросов.

2.2.2 Реализация

1. Рассмотрим точность модели по кросс-валидации с 3 фолдами для каждого k (число ближайших соседей, используемое в алгоритме) от 1 до 10 для каждой из меры близости объектов: евклидового (*euclidean*) и косинусного (*cosine*) расстояний. (См. рис. 2)

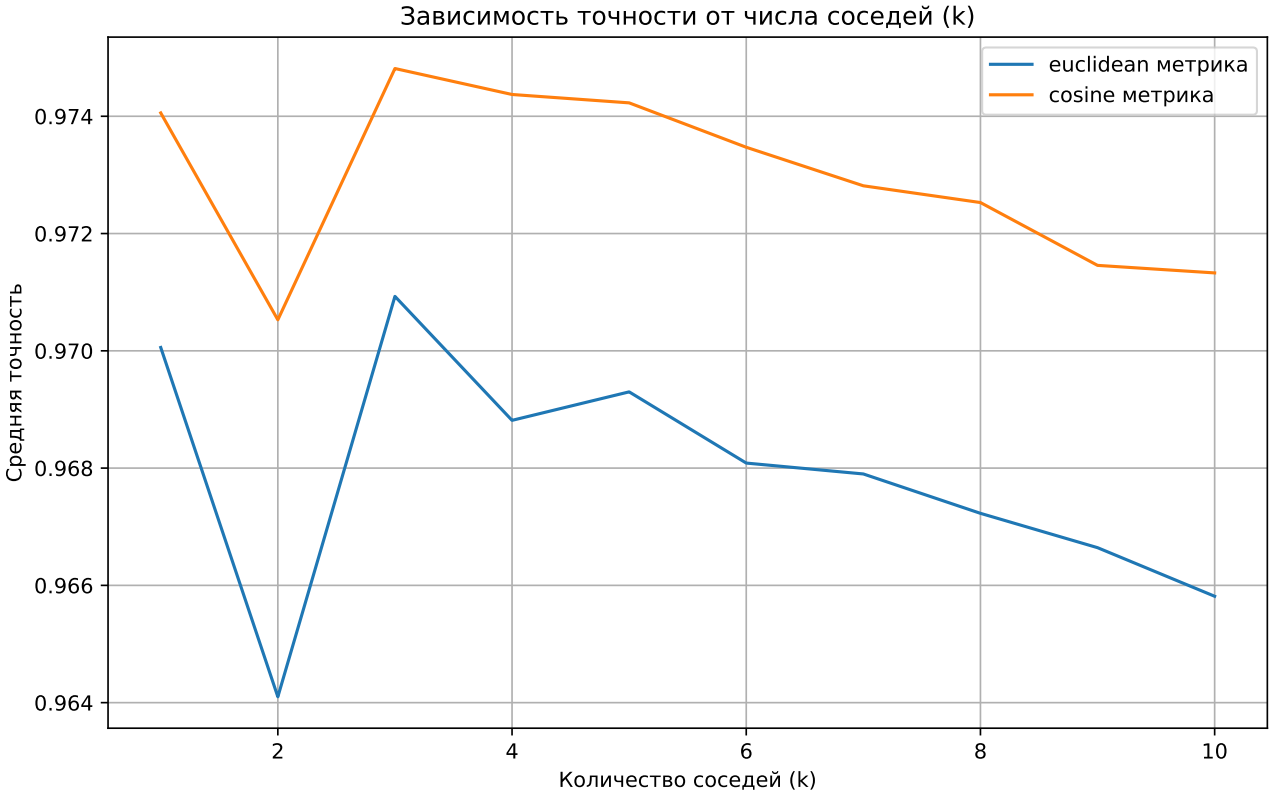

Рис. 2: Точность модели поиска ближайших соседей в заисимости от меры расстояния и k

Как можно заметить из графика, точность модели при любом k для косинусного расстояния больше, чем соответствующие значения точности при использовании евклидового расстояния, что может быть связано с тем, что косинусное расстояние лучше работает с разряженными векторами и является менее чувствительным к смещениям и поворатам изображения. Также, заметно существенное падение при k=2 (числе соседей), т.к. при данном количестве соседей затрудняется выбор целевого класса для объекта выборки, т.к. сложно сделать выбор к какому классу отнести объект, если 1 из 2х соседей относится к одному классу, а другой - к другому.

2. Рассмотрим время выполнения кросс-валидации модели с 3 фолдами для

каждого k от 1 до 10 для каждой из меры близости объектов: (См. табл. 1)

metric	Время выполнения(сек.)
cosine	69.960669
euclidean	30.540650

Таблица 1: Время работы кросс-валидации в зависимости от метрики

Из таблицы видно, что модель, использующая косинусное расстояние, работает дольше, чем KNNClassifier, использующий евклидово расстояние. Это может быть связано со сложностью вычисления косинусного расстояния, т.к. необходимо не только посчитать скалярное произведение каждой пары векторов, но и нормы этих векторов.

2.2.3 Выводы

Исследование показало, что хоть и модель, использующая косинусное расстояние работает дольше той, что использует евклидово, она выигрывает в показаниях точности.

2.3 Сравнение взвешенного метода k ближайших соседей с методом без весов

2.3.1 Постановка задачи

Сравнить взвешенный метод k ближайших соседей, где голос объекта равен $1/(\text{distance} + \varepsilon)$, где $\varepsilon = 10^{-5}$, с методом без весов при тех же фолдах и параметрах.

2.3.2 Реализация

Рассмотрим зависимость точности модели от наличия весов, количества соседей (параметр k) и меры расстояния(*cosine* и *euclidean*): (См. рис. 3 и рис. 4)

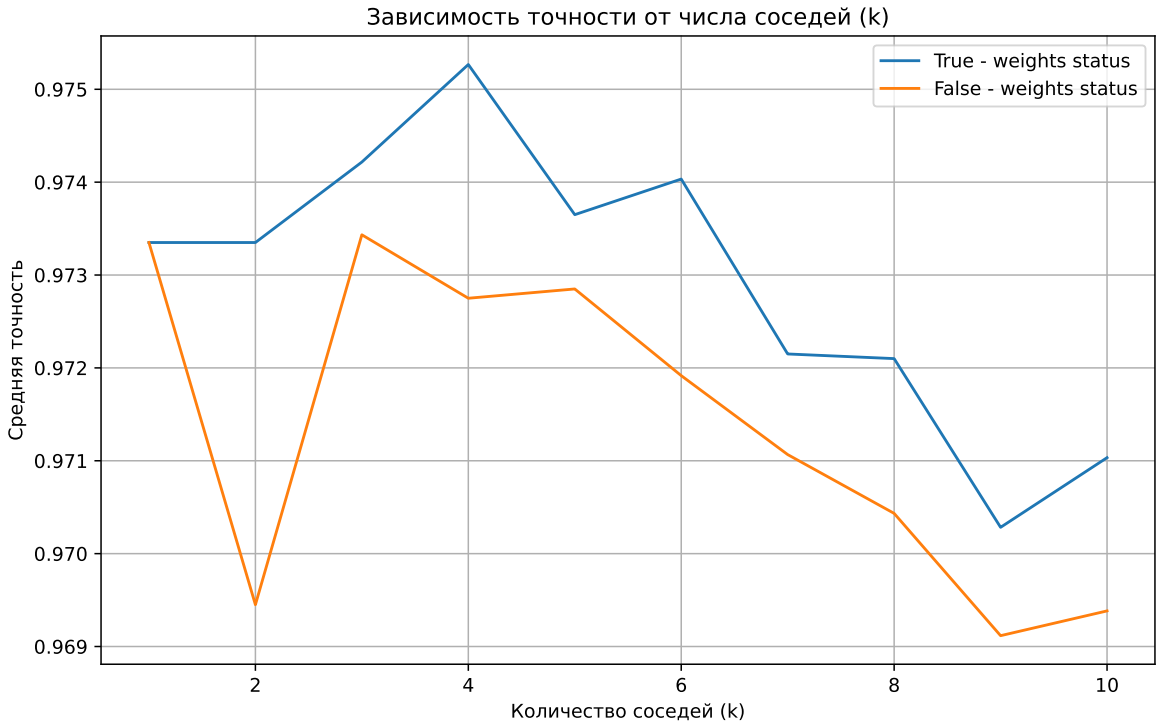

Рис. 3: Зависимость точности от числа соседей для cosine расстояния

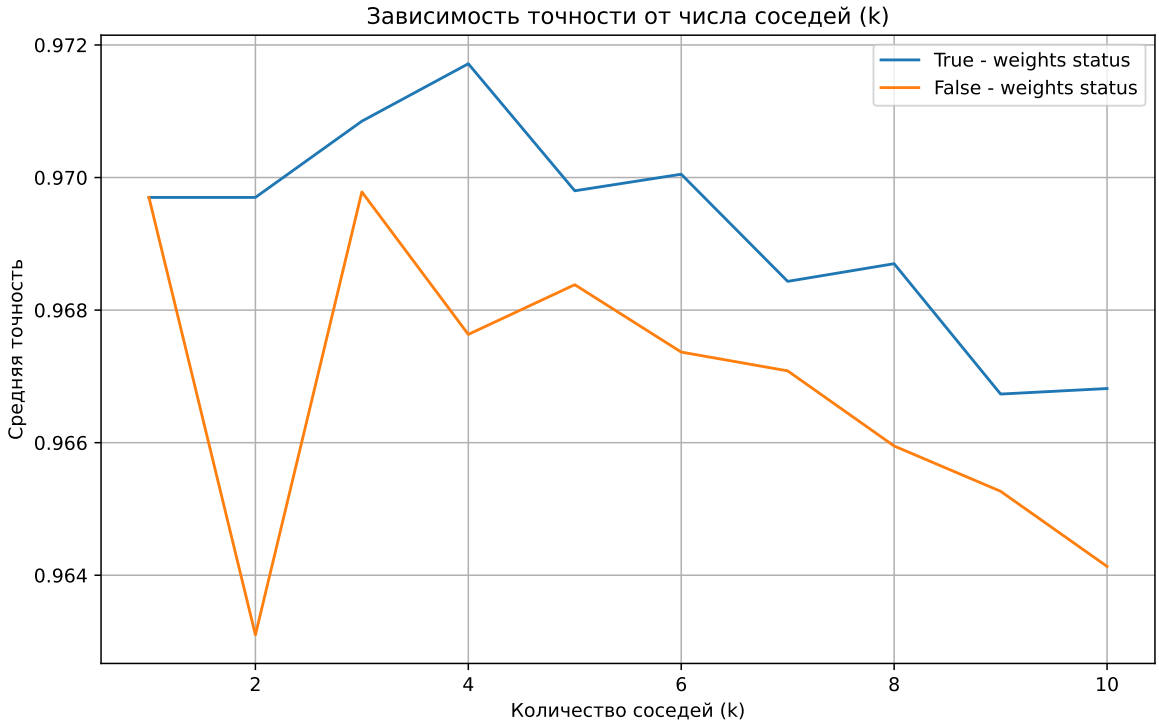

Рис. 4: Зависимость точности от числа соседей для euclidean расстояния

Сравним скорости работы взвешенного метода k ближайших соседей с методом без весов с различными мерами расстояний по кросс-валидации: (См. табл. 2)

metric	weights	Время выполнения(сек.)
euclidean	True	24.149072
euclidean	False	23.085120
cosine	True	48.408618
cosine	False	47.752482

Таблица 2: Время работы кросс-валидации в зависимости от наличия весов и метрики

Из таблицы видно, что модель, использующая косинусное расстояние, работает дольше, чем KNNClassifier, использующий евклидово расстояние, аналогично предыдущему эксперименту. Также можно заметить, что время работы моделей с весами увеличивается относительно соответствующих моделей без весов, что связано к дополнительным вычислениями по подсчёту весов.

2.3.3 Выводы

Исследование показало, что хоть и модели, использующие веса работают дольше тех, что их не используют, но они выигрывают в показаниях точности.

2.4 Исследование точности алгоритма с лучшими параметрами, полученными ранее

2.4.1 Постановка задачи

Применить лучший алгоритм к исходной обучающей и тестовой выборке. Подсчитать точность. Сравнить с точностью по кросс-валидации. Сравнить с указанной в интернете точностью лучших алгоритмов на данной выборке. Выполнить анализ ошибок построением и анализом матрицы ошибок (confusion matrix). Визуализировать несколько объектов из тестовой выборки, на которых были допущены ошибки. Проанализировать и указать их общие черты.

2.4.2 Реализация

По результатам предыдущих экспериментов на данный момент наилучшими параметрами для KNNClassifier являются: metric=*cosine*, weights=*True*, k=*4*. Применим модель с данными параметрами к обучающей выборке и подсчитаем точность на тестовой выборке, а также подсчитаем точность на кросс-валидации на 3х фолдах: (См. табл. 3)

Модель	accuracy
На тестовой	0.975200
Кросс-валидация	0.976757
Лучшая	0.9977

Таблица 3: Точность модели

Из таблицы видно, что модель, полученная при помощи лучших на данный момент параметров проигрывает в точности лучшей в мире модели на 0,0225% точности. Также данная модель по точности хуже среднего значения точности по кросс-валидации, т.к. на кросс валидации модель использовала только 3 фолда. При их увеличениии точность модели на кросс-валидации уменьшится.

Исследуем матрицу ошибок модели, полученную при помощи *sklearn.metrcics.confusion_matrix*: (См. рис. 5)

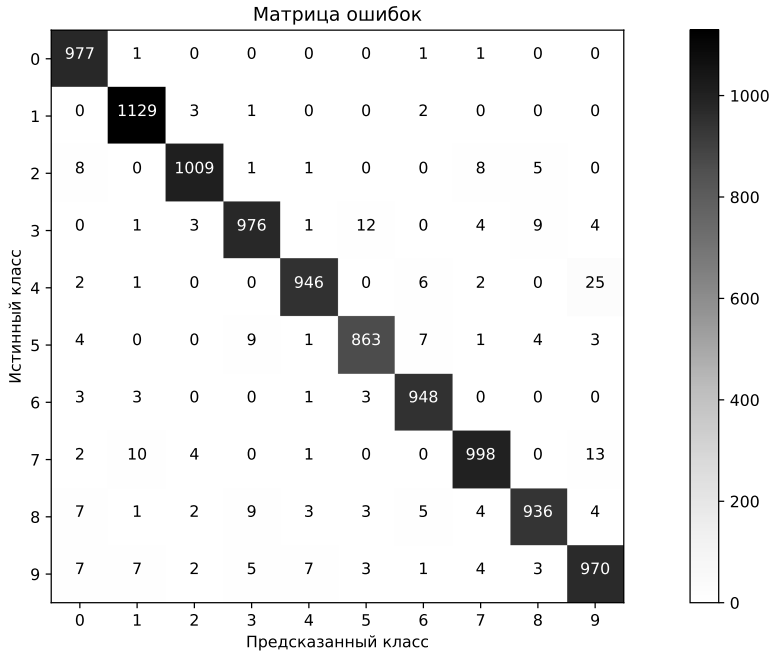

Рис. 5: Матрица ошибок лучшей модели

Рассмотрим популярные ошибки полученной модели(всего ошибок - 248):

Рис. 6: Ошибки по 0

Рис. 7: Ошибки по 1

Рис. 8: Ошибки по 2

Рис. 9: Ошибки по 3

Рис. 10: Ошибки по 4

Рис. 11: Ошибки по 5

Рис. 12: Ошибки по 6

Рис. 13: Ошибки по 7

Рис. 14: Ошибки по 8

Рис. 15: Ошибки по 9

2.4.3 Выводы

Исследование показало, что модель чаще всего путает 3 с 5, 4 с 9, 7 с 9, 7 с 1. По изображениям видно, что из-за рукописного формата цифр и неточности их написания и происходят ошибки. Также, было выяснено, что на наилучшем алгоритме, полученном на данный момент, точность отличается от наилучшего в мире на 0,0225.

2.5 Исследование аугментации обучающей выборки

2.5.1 Постановка задачи

Выполнить аугментацию обучающей выборки. Размножить ее с помощью поворотов, смещений, морфологических операций и применений гауссовского фильтра. Подобрать по кросс-валидации с 3 фолдами параметры преобразований. Рассмотреть следующие параметры для преобразований и их комбинации:

1. Величина поворота: 5, 10, 15 (в каждую из двух сторон)
2. Величина смещения: 1, 2, 3 пикселя (по каждой из двух размерностей)
3. Дисперсия фильтра Гаусса: 0.5, 1, 1.5
4. Морфологические операции: эрозия, дилатация, открытие, закрытие с ядром

Проанализируйте, как изменилась матрица ошибок, какие ошибки алгоритма помогает исправить каждое преобразование.

2.5.2 Реализация

Выполним преобразования:

1. Величина поворота: 5, 10, 15 (в каждую из двух сторон)
   (a) Рассмотрим пример поворота изображений: (См. рис. 16)

Рис. 16: Примеры поворота цифры

(b) Исследуем значение точности модели на полученненных данных: (См. рис. 17)

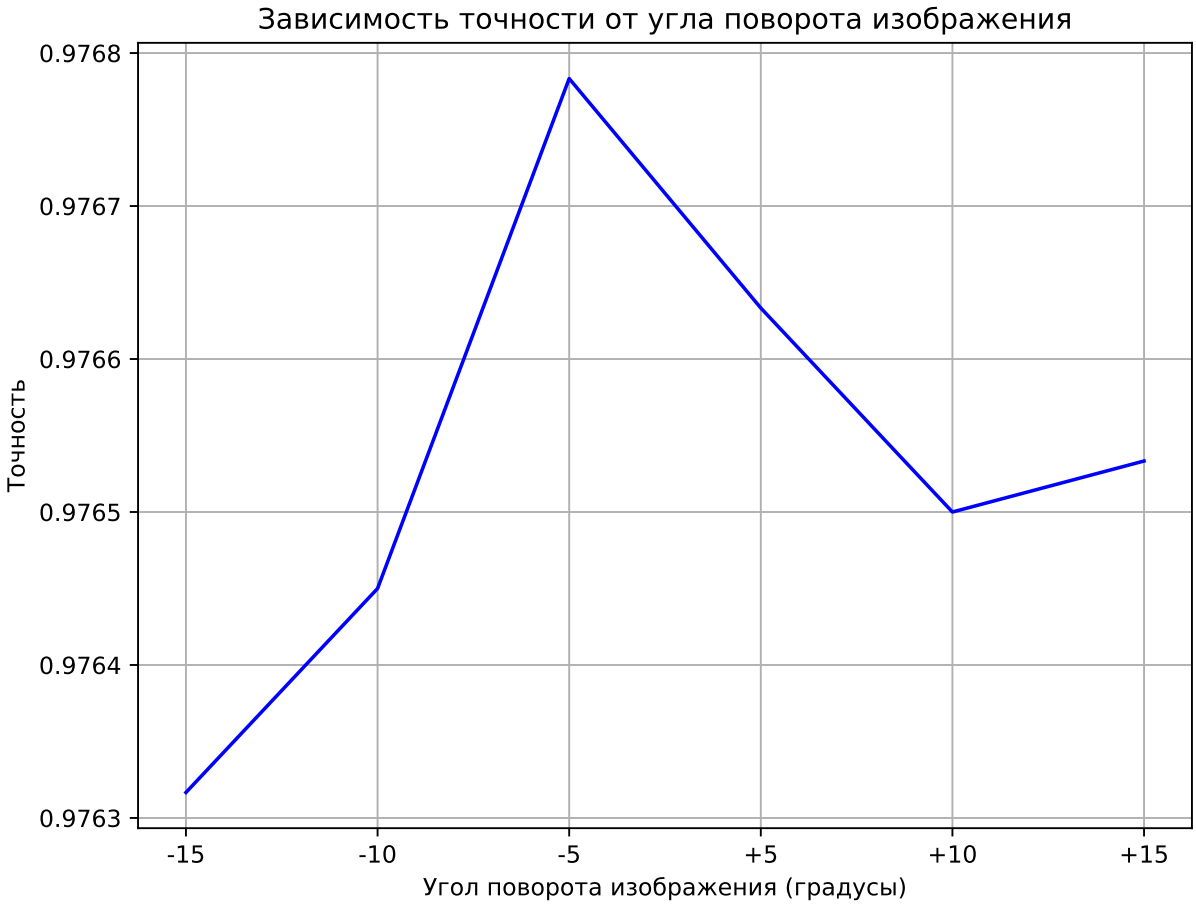

Рис. 17: Точности модели при модификации поворотом

Как видно из графика, наилучший показатель точности был достигнут по кросс-валидации при повороте на 5 градусов вправо.

Исследуем матрицу ошибок модели, обученной на данных, подвергнутых такой модификации: (См. рис. 18)

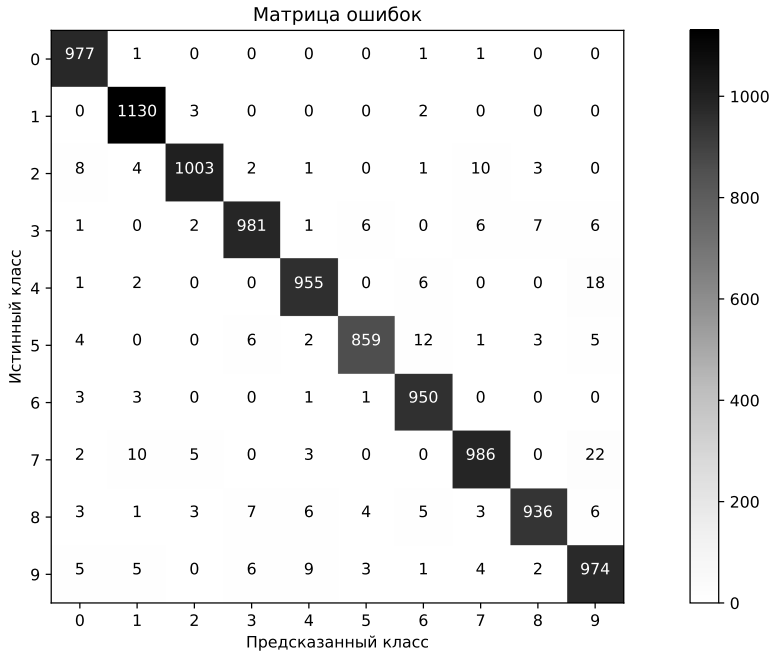

Рис. 18: Матрица ошибок для лучшей модификации поворотом

(c) **Вывод:** можно заметить, что ошибки, связанные со спутыванием 3 с 5 значительно уменьшились.

2. Величина смещения: 1, 2, 3 пикселя по оси OX

(a) Рассмотрим пример смещения изображений по OX: (См. рис. 19)

Рис. 19: Примеры свдига цифры по OX

(b) Исследуем значение точности модели на полученненных данных: (См. рис. 20)

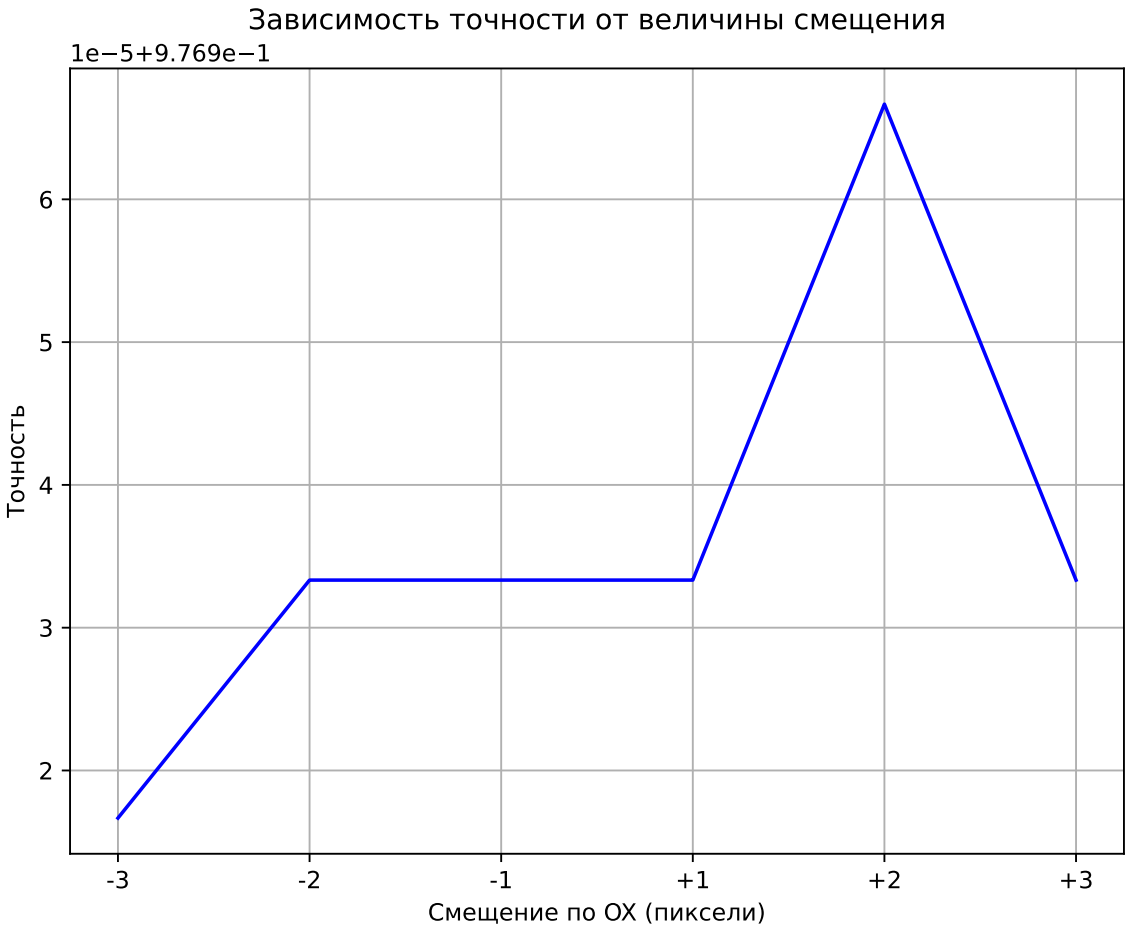

Рис. 20: Точности модели при модифакации сдвигом по OX

Как видно из графика, наилучший показатель точности по кросс-валидации был достигнут при смещении на 2 пикселя вдоль положительного направления оси OX (вправо).

Исследуем матрицу ошибок модели, обученной на данных, подвергнутых такой модификации: (См. рис. 21)

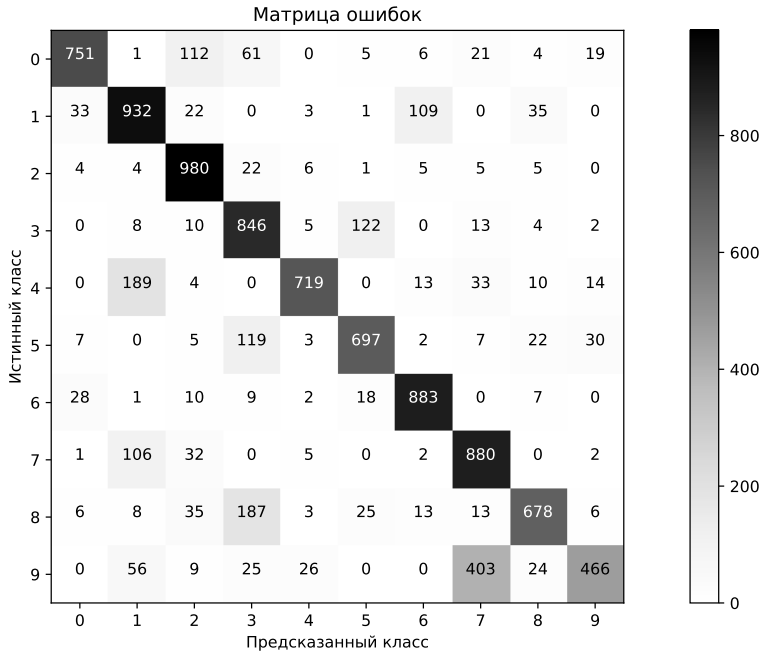

Рис. 21: Матрица ошибок для лучшей модификации сдвигом по OX

(c) **Вывод:** можно заметить, что количество отдельных ошибок наоборот увеличилось при такой модификации.

3. Величина смещения: 1, 2, 3 пикселя по оси OY

(a) Рассмотрим пример смещения изображений по OY: (См. рис. 22)

Рис. 22: Примеры свдига цифры по OY

(b) Исследуем значение точности модели на полученненных данных: (См. рис. 23)

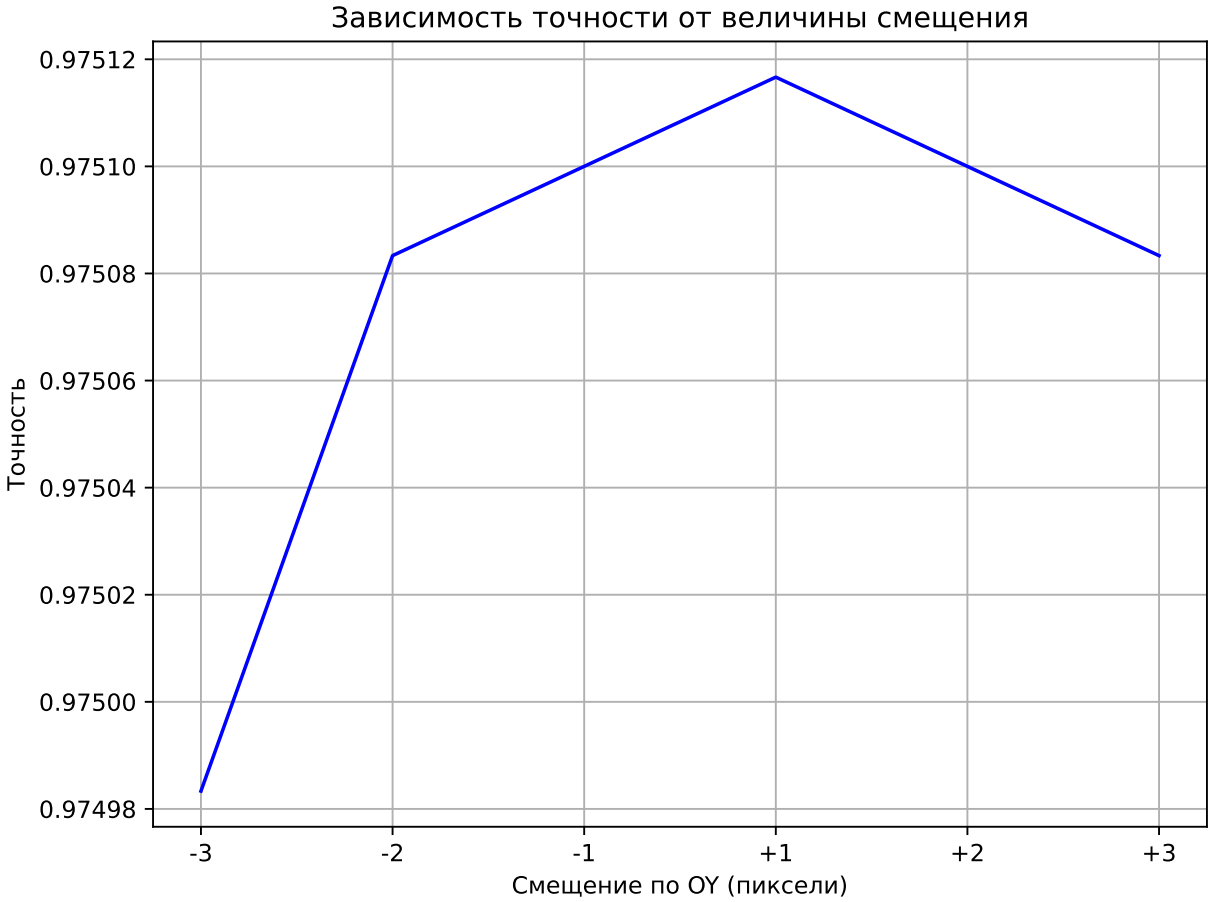

Рис. 23: Точности модели при модификации сдвигом по OY

Как видно из графика, наилучший показатель точности по кросс-валидации был достигнут при смещении на 1 пиксель вдоль положительного направления оси OY (вверх).

Исследуем матрицу ошибок модели, обученной на данных, подвергнутых такой модификации: (См. рис. 24)

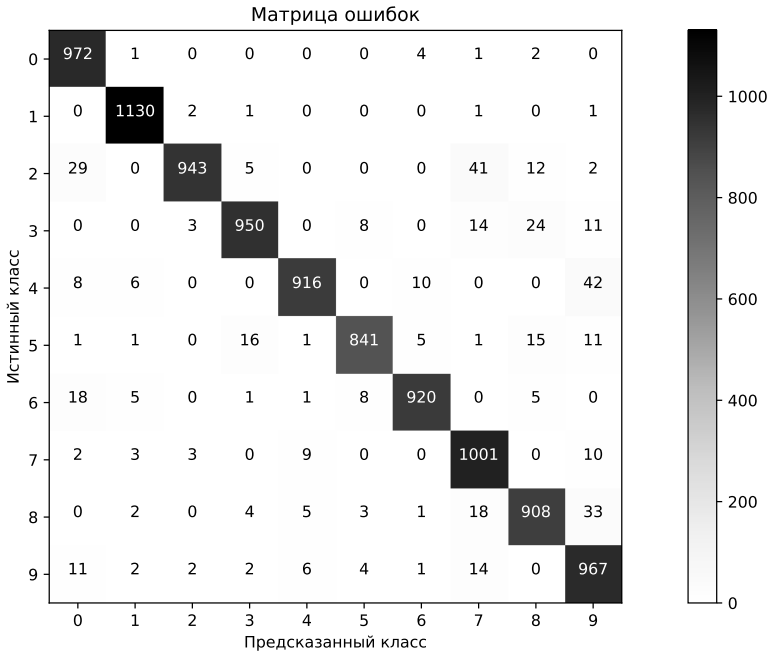

Рис. 24: Матрица ошибок для лучшей модификации сдвигом по OY

(c) **Вывод:** можно заметить, что количество отдельных ошибок наоборот увеличилось при такой модификации, но количество ошибок, связанных со спутыванием 8 и 3, 7 и 9 уменьшилось.

4. Дисперсия фильтра Гаусса: 0.5, 1, 1.5

(a) Рассмотрим примеры дисперсии фильтра Гаусса(0.5, 1, 1.5): (См. рис. 25)

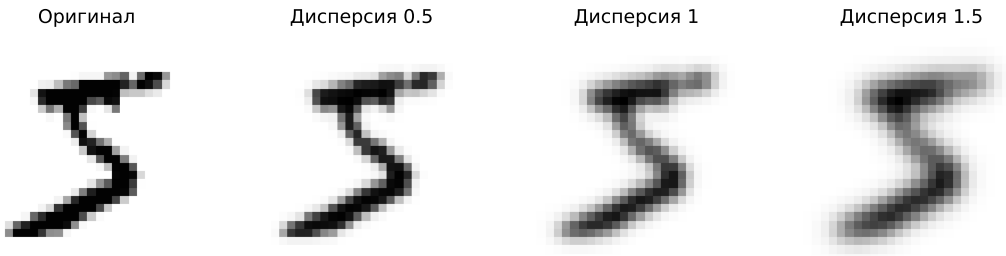

Рис. 25: Примеры дисперсии фильтра Гаусса

(b) Исследуем значение точности модели на полученненных данных: (См. рис. 26)

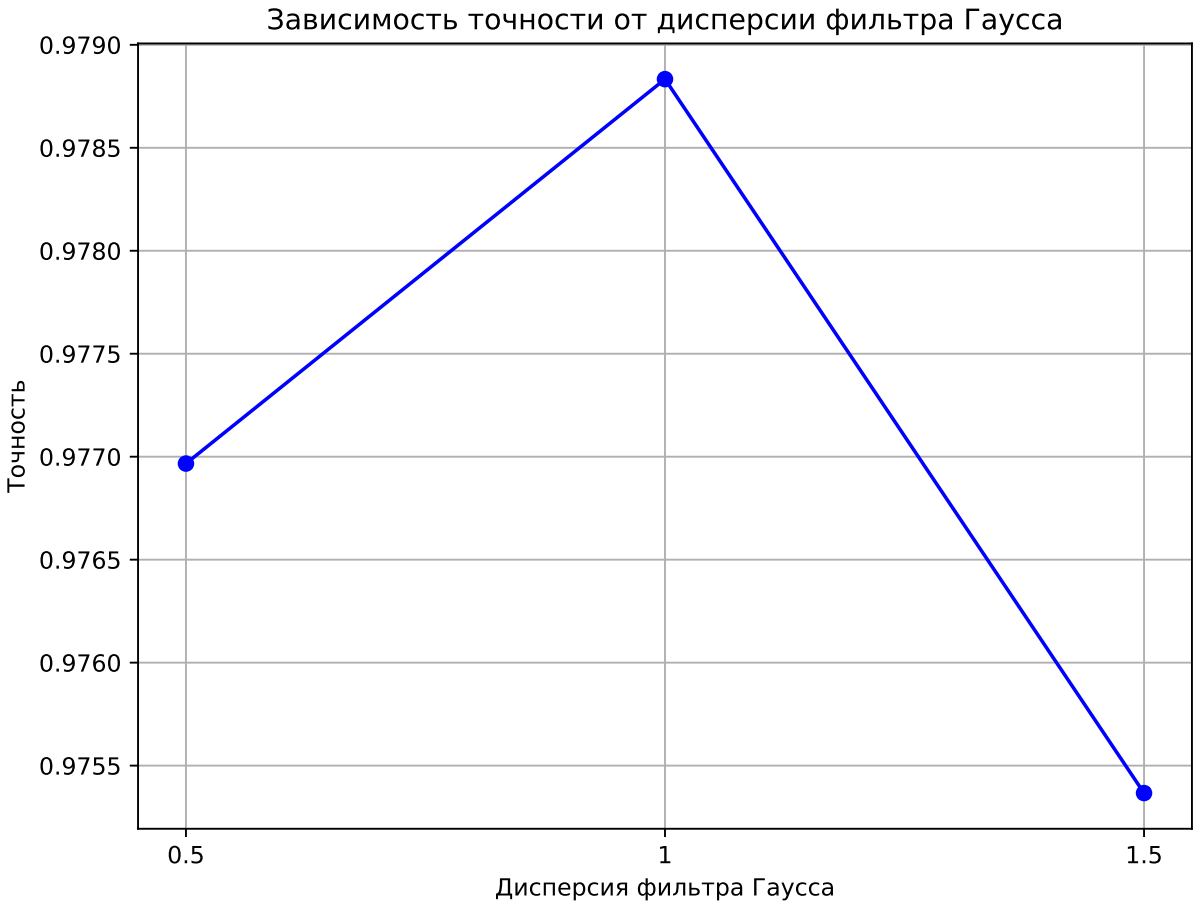

Рис. 26: Точности модели при модификации фильтром Гаусса

Как видно из графика, наилучший показатель точности по кросс-валидации был достигнут при дисперсии фильтра Гаусса равной 1.

Исследуем матрицу ошибок модели, обученной на данных, подвергнутых такой модификации: (См. рис. 27)

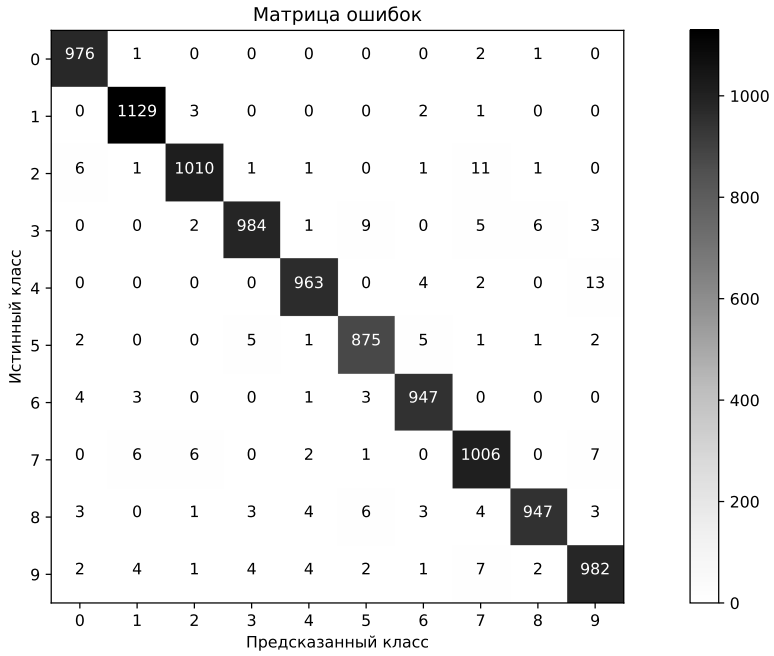

Рис. 27: Матрица ошибок для лучшей модификации фильтром Гаусса

(c) **Вывод:** можно заметить, что количество отдельных ошибок со спутыванием 4 и 9, 6 и 9, 7 и 9 и 7 и 1 уменьшилось.

5. Морфологические операции: эрозия, дилатация, открытие, закрытие с ядром

   (a) Рассмотрим пример эрозии с ядром из единиц размера 2*2: (См. рис. 28)

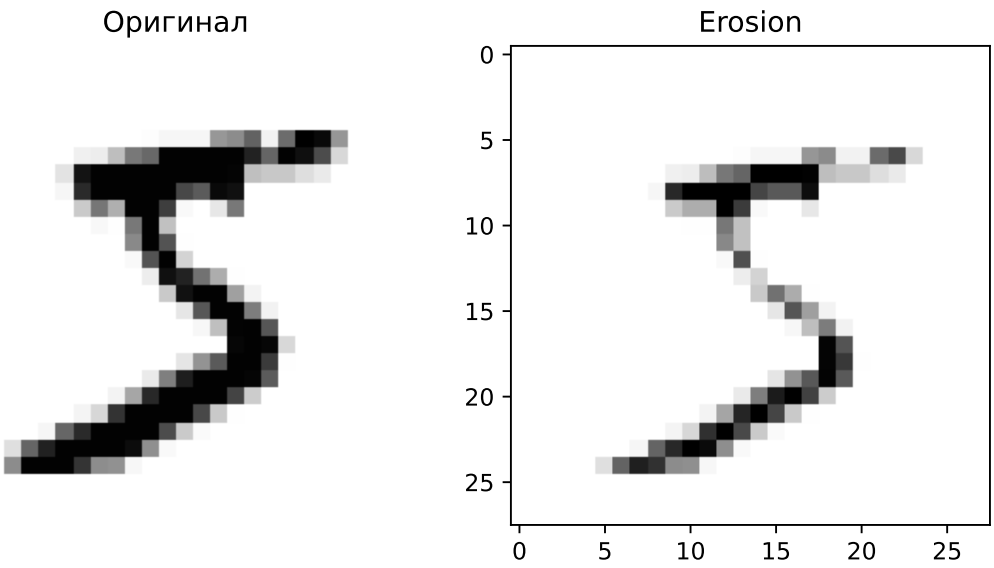

Рис. 28: Пример изображений с эрозией

(b) Рассмотрим пример дилатации с ядром из единиц размера 2*2: (См. рис. 29)

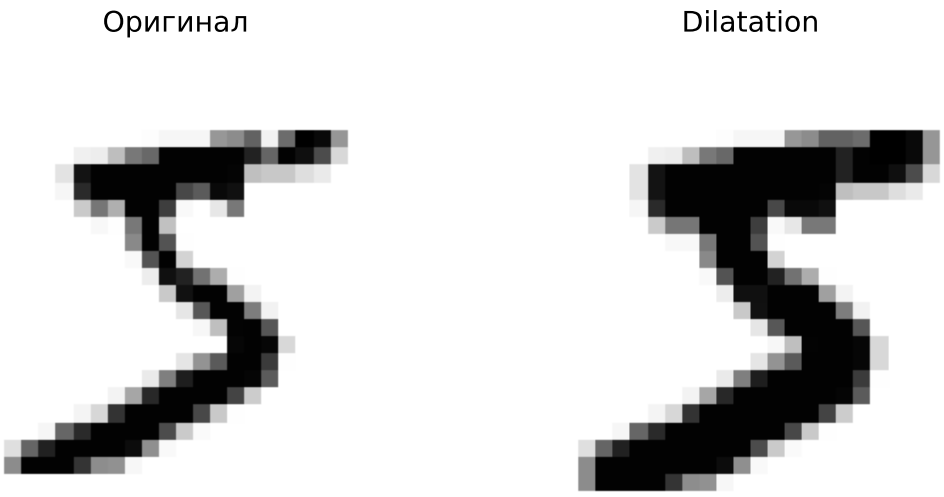

Рис. 29: Пример изображений с дилатацией

(c) Рассмотрим пример открытия с ядром из единиц размера 2*2: (См. рис. 30)

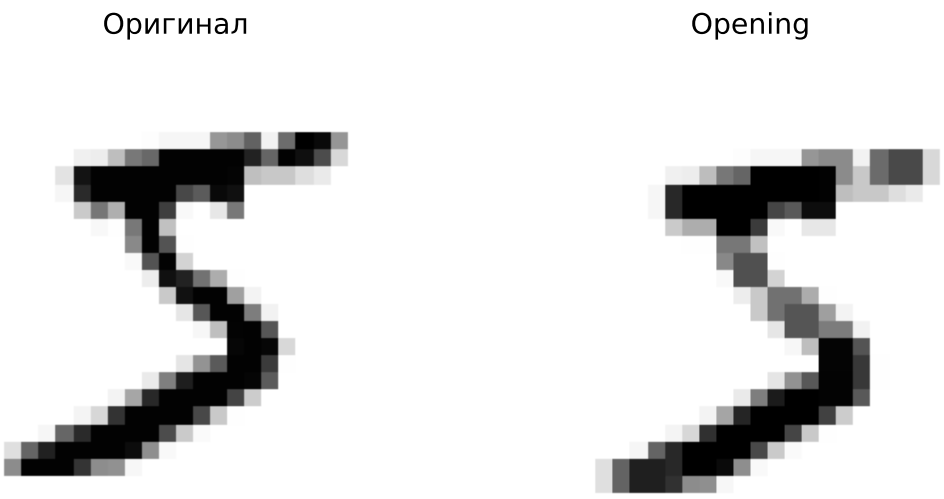

Рис. 30: Пример изображений с открытием

(d) Рассмотрим пример закрытия с ядром из единиц размера 2*2: (См. рис. 31)

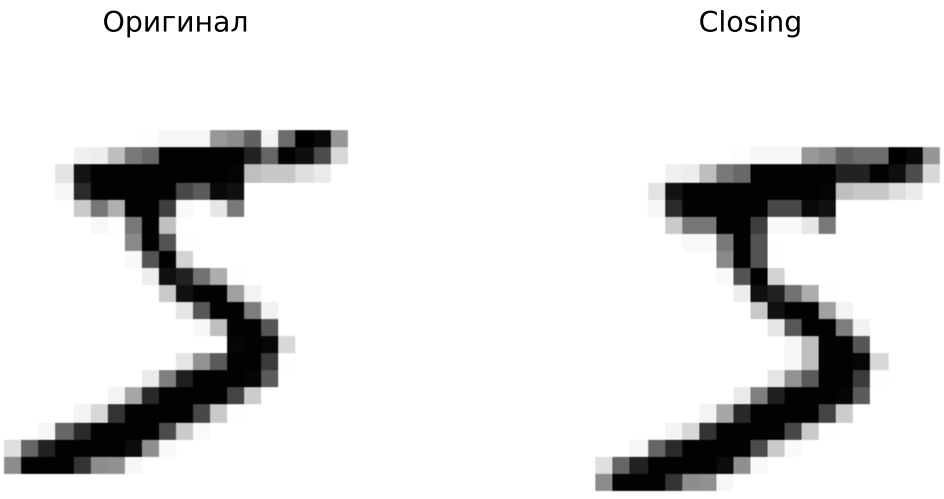

Рис. 31: Пример изображений с закрытием

(e) Исследуем значение точности модели на полученненных данных: (См. рис. 32)

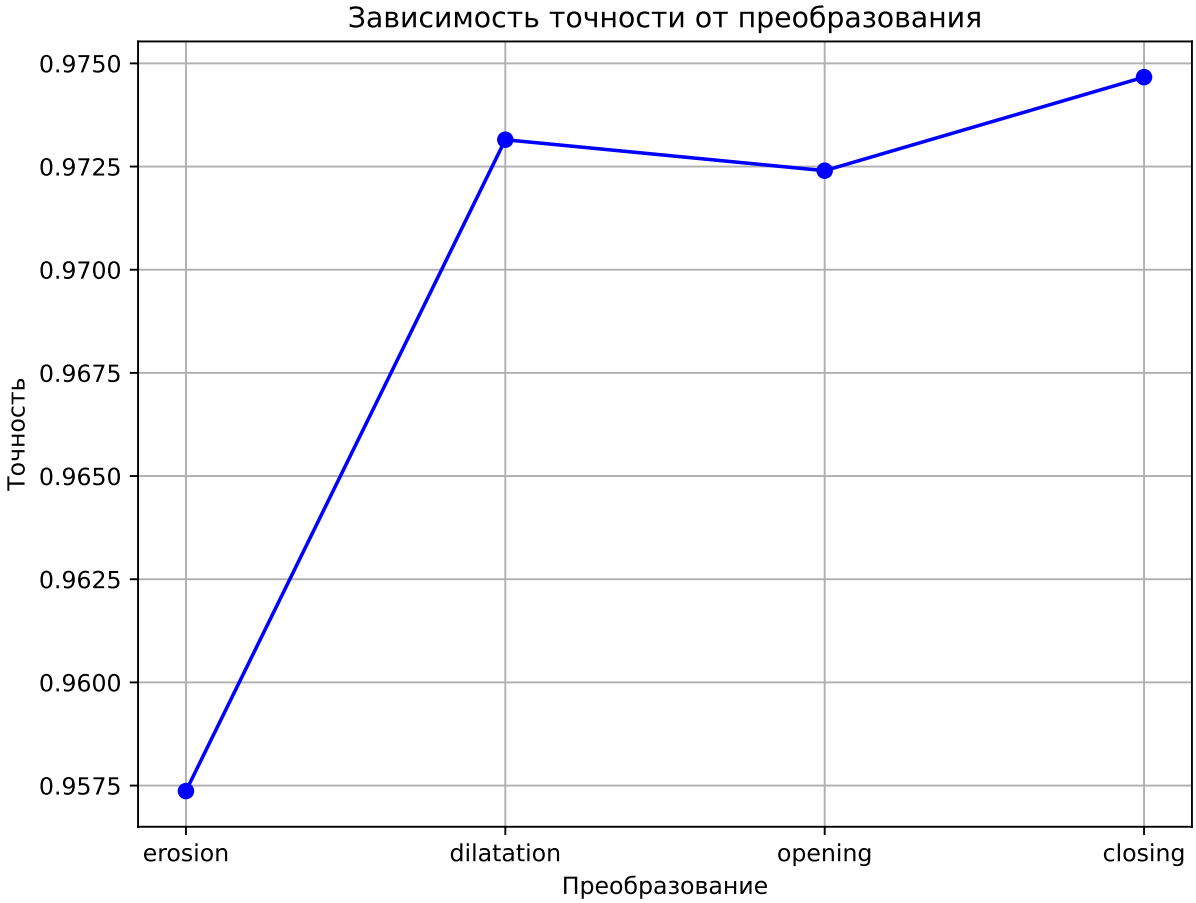

Рис. 32: Точности модели при модификации различными методами

Как видно из графика, наилучший показатель точности по кросс-валидации был достигнут при закрытии ядром из единиц размером 2*2.

Исследуем матрицу ошибок модели, обученной на данных, подвергнутых такой модификации: (См. рис. 33)

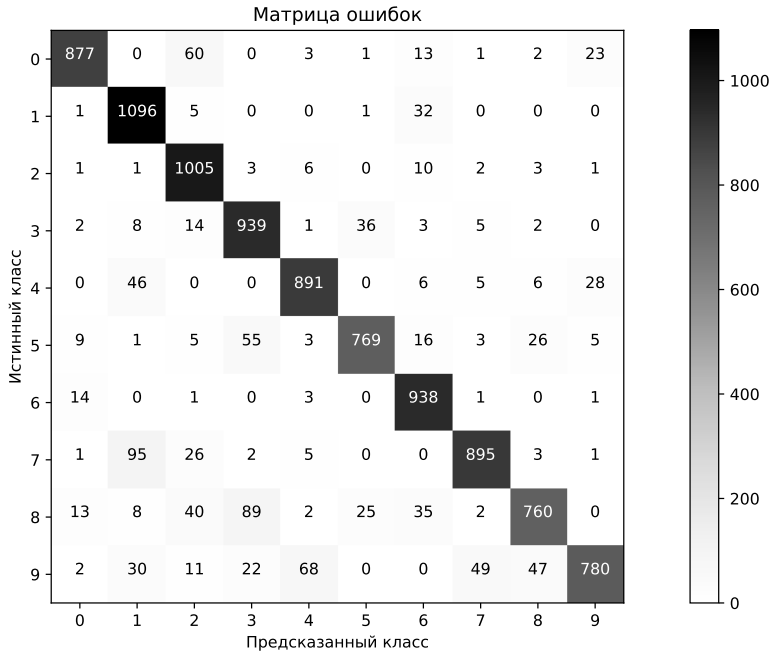

Рис. 33: Матрица ошибок для лучшей модификации

(f) **Вывод:** можно заметить, что количество отдельных ошибок увеличилось, например: 0 с 2, 7 с 1, 8 с 3, 9 с 8.

6. Составление лучшей модели по аугментации

   Итого лучшими в каждой категории по точности стали: поворот на 5 градусов по часовой стрелке, сдвиг по OX на 2 пикселя вправо, сдвиг по OY на 1 пиксель вверх, дисперсия фильтра Гаусса равная 1, закрытие с ядром из единиц размера 2*2. Обучим модель на этих модифицированных данных и исследуем значение точности, а также составим матрицу ошибок.

   Полученная точность на тестовой выборке: **0.983100**

   Рассмотрим матрицу ошибок, получившейся модели: (См. рис. 34)

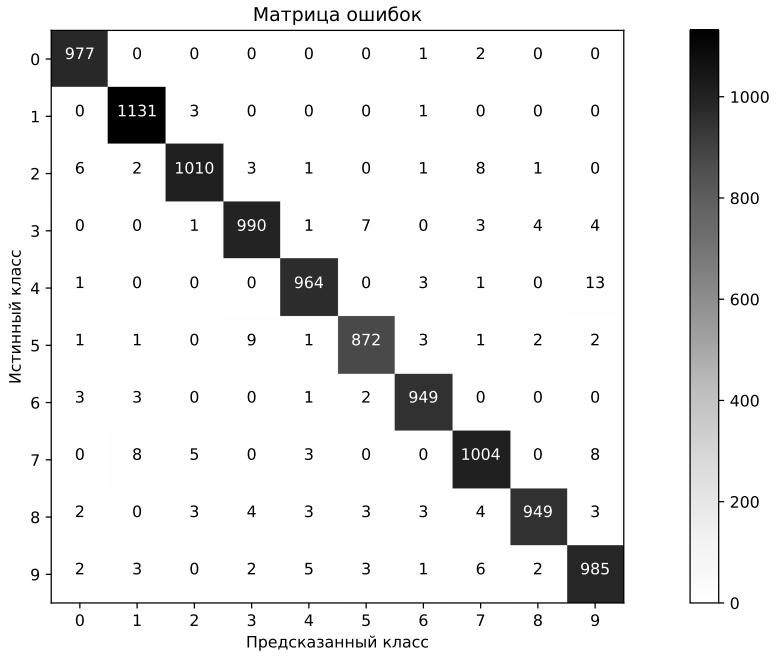

Рис. 34: Матрица ошибок для модели, полученной аугментацией

2.5.3 Выводы

Таким образом, получилась модель с наилучшим показателем точности на данный момент, т.к. она была обучена на аугментированной тестовой выборке суммарным размером 360 тыс. элементов, что улучшило ее точность примерно на 0.01. Также, исследование матрицы ошибок показало, что, в среднем, количество ошибок для каждой категории уменьшилось примерно вдвое.

2.6 Исследование аугментации тестовой выборки

2.6.1 Постановка задачи

Реализовать описанный выше алгоритм, основанный на преобразовании объектов тестовой выборки. Проверить то же самое множество параметров, что и в предыдущем пункте. Проанализировать как изменилась матрица ошибок, какие ошибки алгоритма помогает исправить каждое преобразование. Качественно сравнить два подхода (5 и 6 пункты) между собой.

2.6.2 Реализация

Ананлогично 5 пункту, для обучения будет использоваться модель KNNClassifier со следующими параметрами: metric=*cosine*, weights=*True*, k=*4*.

Модифицируем и размножим тестовую выборку указанными ранее способами. Всего получилось - 19 параметров. Новая тестовая выборка будет иметь размер

190 тыс. элементов. Обучим на обычной обучающей выборке нашу модель и измерим точность модели:

Получившаяся точность на тестовой выборке: **0.891442**.

Также, построим матрицу ошибок для получившейся модели: (См. рис. 35)

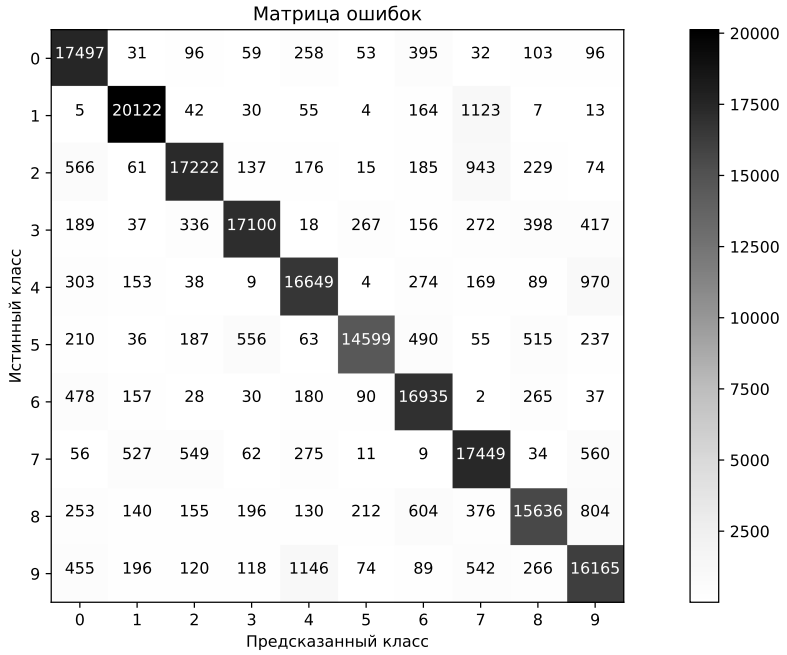

Рис. 35: Матрица ошибок для полученной модели

Из матрицы заметно, что количество ошибок возросло в разы для каждой из возможной пары "предсказанное значение "истинное значение".

2.6.3 Выводы

Исследование показало, что такой подход только ухудшает результат и не даёт прироста точности. Аугментирование тестовой выборки без обучающей только увеличивает сложность предсказаний для модели. Аугментация тестовой выборки нарушает принцип честной оценки модели и приводит к изменению данных, которые могут сильно отличаться от обучающих, что приводит только к уменьшению значения точности.

2.7 Вывод

В ходе выполнения данного задания был рассмотрен метричесие методы машинного обучения на примере метода k ближайших соседей в задаче распознавания рукописных цифр из датасета MNIST. Были рассмотрены различные подходы к данной задаче: от более подробного изучения параметров метода KNN и определения лучших для данного датасета, до исследования аугментации изображений и

её анализа. Лучшая модель, полученная обучением на аугментированных данных, показала точность **0.891442**.

Список литературы

[1] opencv, https://docs.opencv.org/3.4/d9/d61.html

[2] kaggle, https://www.kaggle.com/c/digit-recognizer/models