# Отчет о практическом задании «Метрические алгоритмы классификации».

Практикум 317 группы, ММП ВМК МГУ.

Морозов Иван Дмитриевич.

Октябрь 2023.

## Содержание

# 1 Введение

Данное практическое задание посвящено исследованию метрических алгоритмов классификации на примере задачи распознавания рукописных цифр. Целями исследования являются сравнение времени работы различных алгоритмов поиска ближайших соседей, сравнение точности и времени работы для различных метрик, влияние на точность и время работы алгоритма использования взвешенного метода ближайших соседей, рассмотрение степени влияния аугментации обучающей и тестовой выборок на точность.

# 2 Эксперимент №1

## 2.1 Постановка

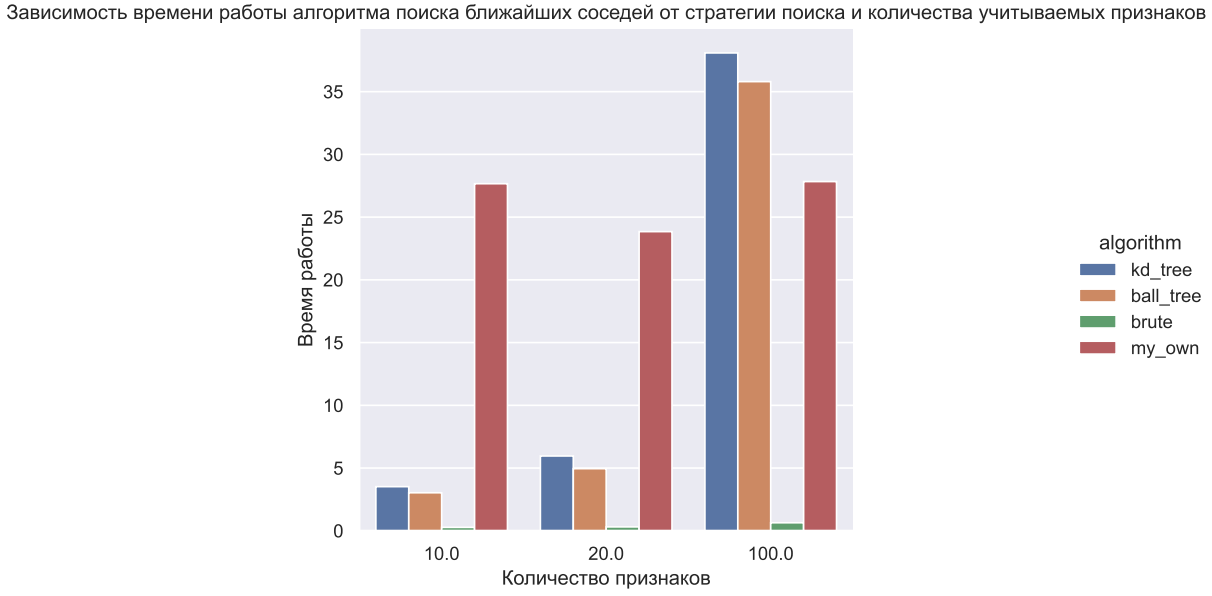
Эксперимент направлен на исследование зависимости времени работы алгоритма поиска ближайших соседей от стратегии поиска и количества учитываемых признаков.

Используется евклидова метрика. Производится перебор всех возможных комбинаций стратегий поиска и количества признаков. Стратегия поиска могут принимать следующие значения: `kd_tree`, `ball_tree`, `brute`, `my_own`, количество признаков: 10, 20, 30. Для каждой пары значений запускается счётчик времени, создается новый экземпляр классификатора, обучается на обучающей выборке, используя нужное количество признаков. Для каждого объекта обучающей выборки выбираются первые `count` признаков, т. е. не случайным образом, поскольку в этом эксперименте оценивается время работы, а не качество модели. Далее для каждого объекта обучающей выборки классификатор ищет 5 ближайших соседей, счётчик остановливается. Данные о паре значений сохраняются.

По данным о каждой паре значений строится график зависимости времени работы от стратегии поиска и количества признаков.

## 2.2 Результаты

Результаты эксперимента представлены на графике.




Время работы алгоритмов поиска `kd_tree`, `ball_tree` возрастает при увеличении количества признаков до 100. Стратегия поиска `my_own` использует примерно одинаковое время для нахождения 5 ближайших соседей для любого из рассмотренных количеств признаков. Время работы `brute` существенно отличается от остальных, этот алгоритм очень быстрый даже для большого количества признаков.

### 2.3 Выводы

`brute` - это самая быстрая стратегия поиска из рассмотренных. Поиск ближайших соседей - это точный алгоритм, поэтому на качество предсказания не будет влиять его реализация. Поэтому в последующих экспериментах будет использоваться стратегия `brute` для ускорения расчётов.

## 3 Эксперименты №2, №3

### 3.1 Постановка

Эксперимент 2 направлен на исследование зависимости точности и времени работы алгоритма поиска `k` ближайших соседей от `k` и используемой метрики.

Эксперимент 3 направлен на выявление различий между взвешенным методом `k` ближайших соседей и методом без весов.

Производится перебор всевозможных комбинаций следующих параметров:

- `k` - количество ближайших соседей, может принимать значения от 1 до 10 включительно.
- `metric` - название используемой метрики. Может быть евклидовой или косинусной.
- `weight` - фактор взвешенности метода. Может принимать значения `True` и `False`.

Для каждой комбинации запускается счётчик времени, производится кросс-валидация с 3 фолдами точности, счётчик останавливается. Сохраняются следующие данные о произведённой итерации кросс-валидации: количество ближайших соседей, название метрики, фактор взвешенности, точность (среднее точности для каждого из фолдов кросс-валидации), затраченное время.

По данным о каждой комбинации строится график зависимости точности от количества ближайших соседей, метрики и фактора взвешенности и график зависимости времени работы от количества ближайших соседей, метрики и фактора взвешенности.

### 3.2 Результаты

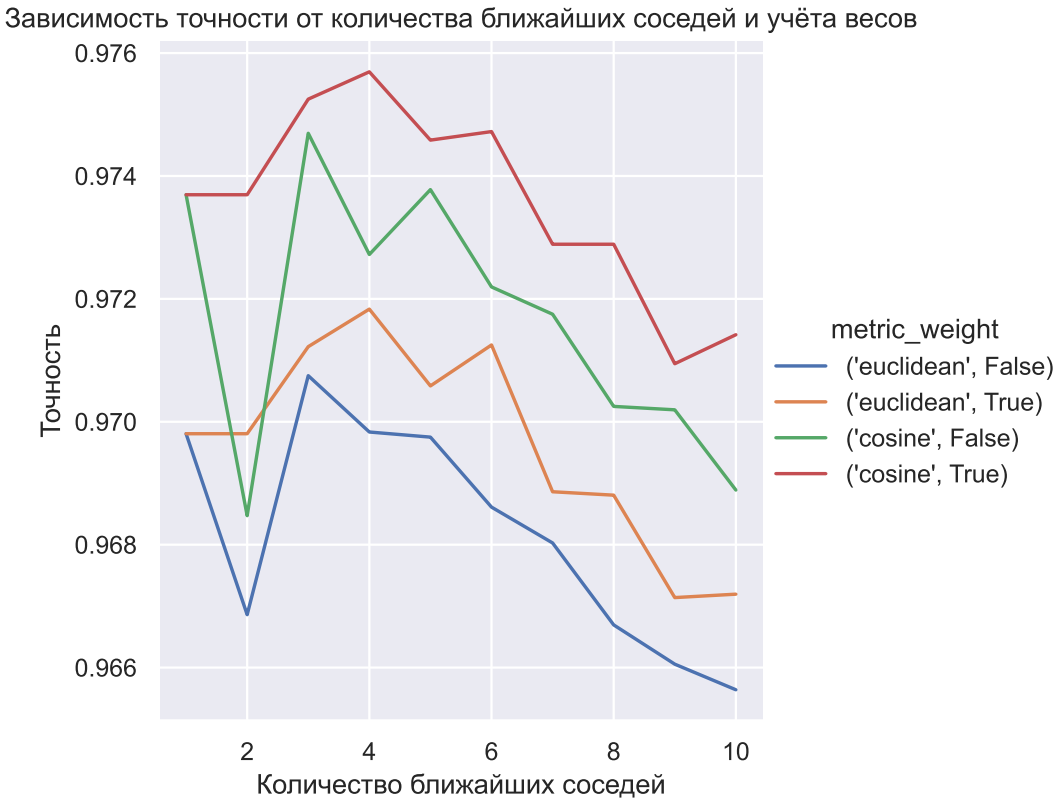
Результаты эксперимента представлены на 2 графиках ниже.

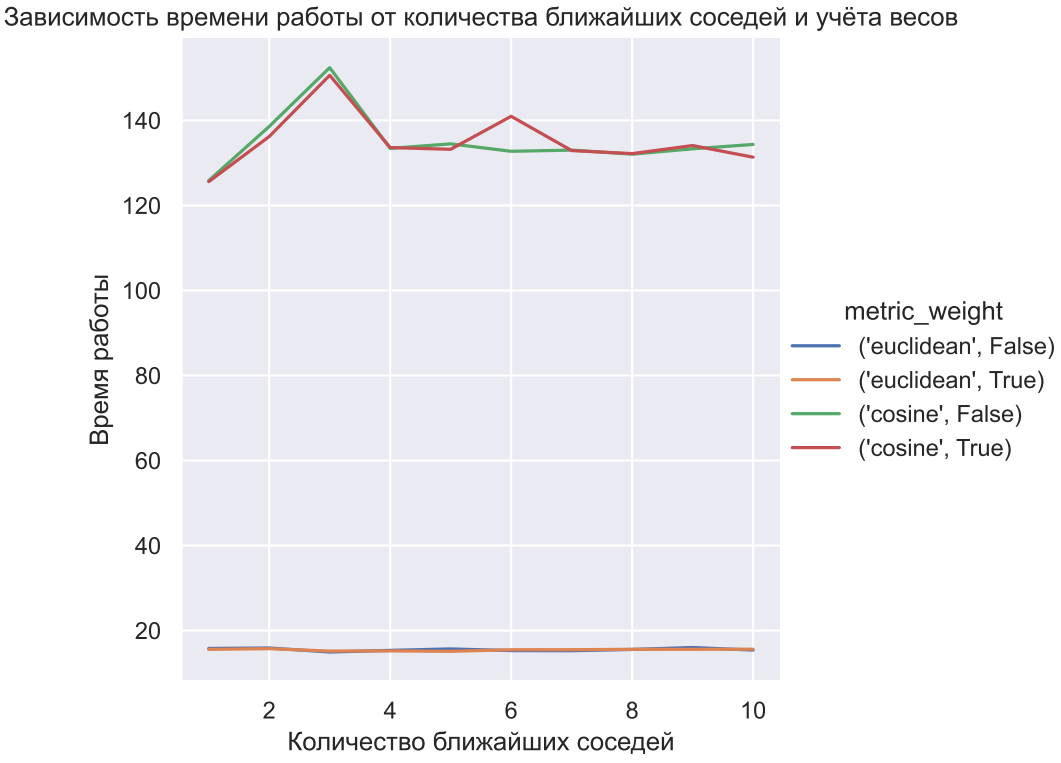
#### 3.2.1 Метод без весов

Точность косинусной метрики лучше точности евклидовой метрики для любого количества ближайших соседей. Наилучшая точность для обеих метрик достигается для 3 ближайших соседей. Когда количество ближайших соседей равно 2, наблюдается значительная просадка точности по сравнению с 1 и 3. С увеличением количества ближайших соседей с 3 до 10 точность постепенно падает. Косинусная метрика затрачивает существенно больше времени, чем евклидова.

#### 3.2.2 Метод с весами

Точность косинусной метрики лучше точности евклидовой метрики для любого значения количества ближайших соседей. Наилучшая точность для обеих метрик достигается для 4 ближайших соседей. Когда количество ближайших соседей равно 2, уже не наблюдается значительных просадок точности, как в случае без весов. С увеличением количества ближайших соседей с 4 до 10 точность постепенно падает. Косинусная метрика затрачивает существенно больше времени, чем евклидова.




### 3.3 Выводы

Для распознавания изображений цифр лучшей метрикой является косинусная, это связано с тем, что цифры - это больше векторные объекты, чем растровые, существует много различных вариантов

написания одной и той же цифры, но ключевую роль играет геометрия и взаимное расположение линий, составлющих образ.

Для метода без весов наилучшее число соседей равно 3, для метода с весами - 4. При таких значениях учитывается наиболее существенная информация о ближайших соседях, не искажаемая более далёкими, т. е. "шумовыми"объектами.

Факт того, что при невзвешенном методе есть просадка точности при 2 ближайших соседях, а при взвешенном её нет, говорит о том, что вторые ближайшие соседи находятся в среднем на более большом расстоянии, чем первые: метод без весов эти расстояния не учитывает и строит прогноз, с равной долей учитывая влияние более далёкого второго соседа и более близкого первого. Метод с весами учитывает и снижает влияние второго соседа.

Косинусная метрика значительно дольше евклидовой. Это связано с более большими объемами вычислений: по реализации расчёта расстояний видно, что в случае косинусной метрики добавляется 2 дополнительных деления.

# 4 Эксперимент №4

## 4.1 Постановка

Эксперимент направлен на обобщение полученных результатов для моделирования лучшего алгоритма, сравнение его с лучшими в мире алгоритмами для данной выборки на основе точности предсказания. Также необходимо визуализировать объекты, на которых алгоритм ошибся и выяснить причины.

Ранее было установлено, что для самой лучшей точности нужно использовать косинусную метрику и искать 4 ближайших соседей. Создается объект классификатора с такими параметрами, обучается на обучающей выборке, строит предсказание для тестовой выборки. По доле правильных ответов строится качество на тестовой выборке. Выполняется кросс-валидация и строится качество на обучающей выборке (среднее точности для каждого из фолдов кросс-валидации).

С помощью библиотеки `sklearn` строится матрица ошибок: строкам соответствуют истинные значения ответов на тестовой выборке, столбцам - предсказанные значения. На пересечении строк и столбцов записывается число, равное количеству объектов с таким истинным значением ответа и прогнозом. Функция `print_conf_matr` рисует матрицу ошибок в виде тепловой карты.

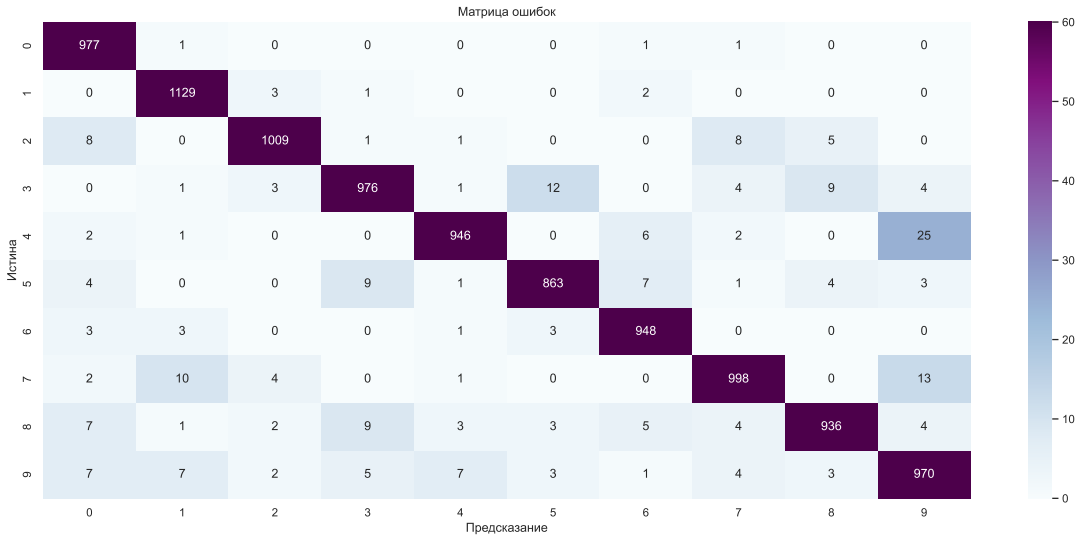
Для визуализации некоторых объектов, на которых произошли ошибки, используется следующий метод. Необходимо отрисовать начертания цифр, на которых алгоритм ошибся. Цикл проходит все прогнозы и истинные значения кругами, находя сначало 10 цифр 0 (истинных), 10 цифр 1 и так далее. С помощью функции `imshow` отрисовываются цифры (на вход подается вектор пикселей (признаков), преобразованный в матрицу 28 на 28).

## 4.2 Результаты

Начертания цифр, на которых произошли ошибки.
Качество по кросс-валидации: 0.9754.
Качество на тестовой выборке: 0.9752.
Качество лучших алгоритмов:

- Система LIPA (трёхслойный перцептрон Розенблатта) 2004 год Качество: 0.9958.
- Нейронные сети 2011 год Качество: 0.9973.
- Свёрточная нейронная сеть Качество: 0.9979.
- RMDL 2018 год Качество: 0.9982.

Начертания цифр, на которых произошли ошибки.


## 4.3 Выводы

Точность по кросс-валидации оказалась выше, чем на тестовой выборке, но заметно ниже других алгоритмов, использующих нейронные сети.

Анализ матрицы ошибок показывает, что наиболее часто алгоритм ошибается на схожих по начертанию цифрах. Например, ’4’ похоже на ’9’ (25 ошибок), ’7’ похоже на ’9’ (13 ошибок), ’3’ похоже на ’5’ (12 ошибок) и так далее. Однако на явно не похожих цифрах алгоритм вообще ни разу не ошибся: например, ’5’ и ’1’.

Рассмотрим начертания цифр, на которых произошли ошибки. Многие из них имеют очень жирное начертание, что затрудняет алгоритм распознать ось скелета, также многие из них имеют непропорциональные размеры составляющих, что влечет ошибки косинусной метрики, у некоторых цифр слабо

прорисованы некоторые линии, что создает иллюзию для алгоритма, что в таких местах линии вообще нет. Также заметен чрезмерный наклон некотрых цифр. Это, например, приводит к ошибкам на цифрах '1' и '7'.

# 5 Эксперимент №5

## 5.1 Постановка, результаты и выводы

Данный эксперимент направлен на улучшение обучающей выборки путём добавления в неё новых объектов и ответов на них. Эти новые объекты создаются на основе уже существующих путём применения к ним следующих преобразований:

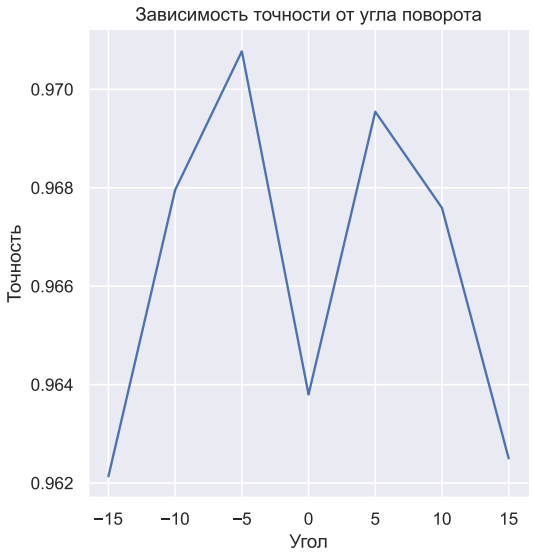
- Поворот
- Смещение
- Фильтр Гаусса
- Эрозия
- Дилатация
- Открытие
- Закрытие

Во всей обучающей выборке `train_X` 60000 элементов. Случайным образом выбираются 20000 элементов. Далее в разделах этого эксперимента рассматривается именно эта выборка (возможно, преобразованная, если преобразование дало улучшение качества - так называемый "жадный"алгоритм). Её название - `aug_train_X`, ответы `aug_train_y`. К ней будет применяться аугментация, и по кросс-валидации будут подобраны оптимальные параметры.

### 5.1.1 Поворот

К выборке `aug_train_X` применяется кросс-валидация с 3 фолдами точности для оценки качества до размножения выборки. Исходное качество: 0.9638. Эти данные сохраняются. Случайным образом выбираются 2000 элементов, их индексы сохраняются, поскольку далее преобразованным объектам нужно сопоставить корректные ответы. Эти 2000 объектов будут непосредственно преобразованы.

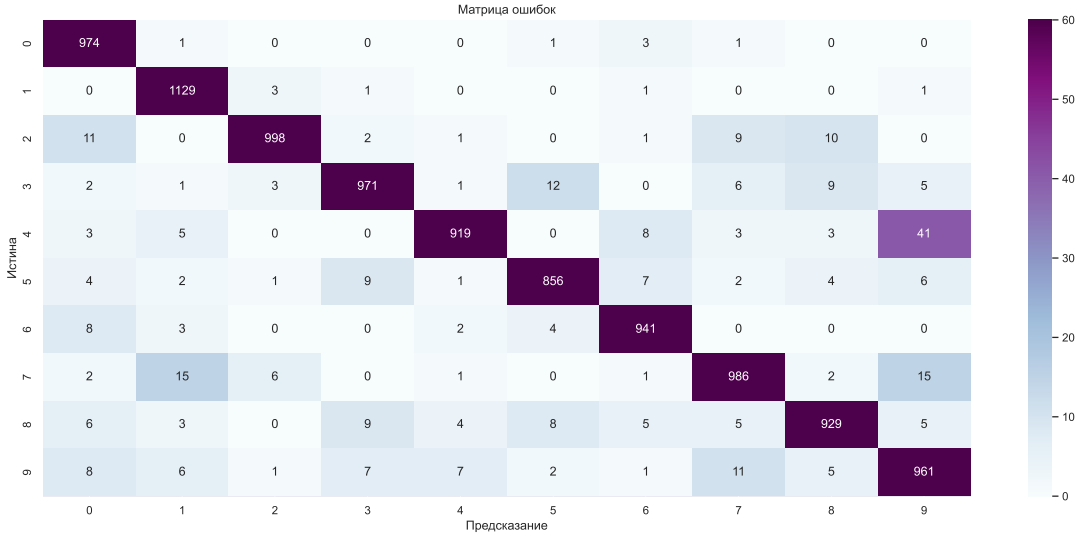
В цикле перебираются все возможные значения угла: 5, 10, 15. Кроме того, берутся как положительные, так и отрицательные значения. Формируется `rotated_train` - выборка, каждый элемент которой получен из исходной выборки (состоит из 2000 элементов) применением функции `skimage.transform.rotate`, поворачивающей изображение на определенный угол. `aug_train_X` конкатенируется с `rotated_train`, ответы, в свою очередь, конкатенируются с 2000 ответами, индексы которых были сохранены ранее. К расширенной выборке применяется кросс-валидация с 3 фолдами точности. Среднее по фолдам качество сохраняется для каждого значения угла. В начале каждой итерации цикла `aug_train_X` и `aug_train_y` остаются неизменными, конкатенация только для кросс-валидации. Строится график зависимости точности от угла поворота. По графику видно, что наибольшее улучшение качества достигается для



угла -5. Таким образом, поворот - полезное преобразование. Если не поворачивать, то на тестовой выборке качество хуже.

Принимается решение добавить `rotated_train` для наилучшего угла в обучающую выборку. Снова строится `rotated_train` для угла -5. Выполняются аналогичные конкатенации для признаков и ответов, но теперь `aug_train_X` и `aug_train_y` изменяются - в них добавляется 20000 элементов. Чтобы размеры `aug_train_X` и `aug_train_y` сделать прежними, случайным образом выбирается 2000 индексов, соответствующие объекты и ответы на них удаляются из `aug_train_X` и `aug_train_y`.

С помощью библиотеки `sklearn` строится матрица ошибок на `aug_train_y`.

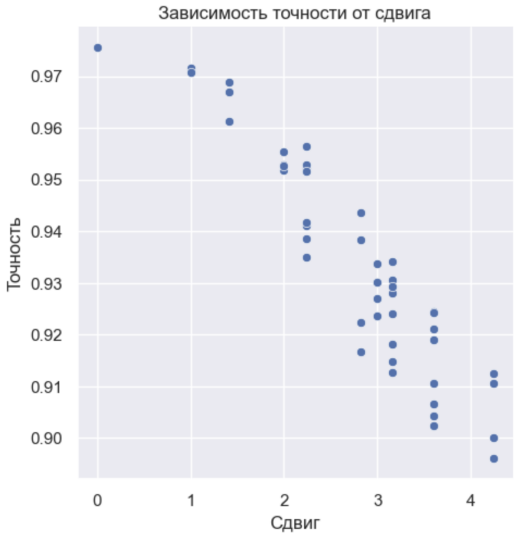



Заметно, что в результате поворота цифра '4' стала чаще путаться с цифрой '9', поскольку '4' имеет больший наклон, чем '9'. Качество на тестовой выборке после аугментации: 0.9664. Качество улучшилось на 0.0026. Таким образом, поворот помогает распознать цифры, написанные под углом и повышает точность.

### 5.1.2 Смещение

Снова случайным образом выбираются 2000 элементов из `aug_train_X`, их индексы сохраняются. Эти 2000 объектов будут непосредственно преобразованы.

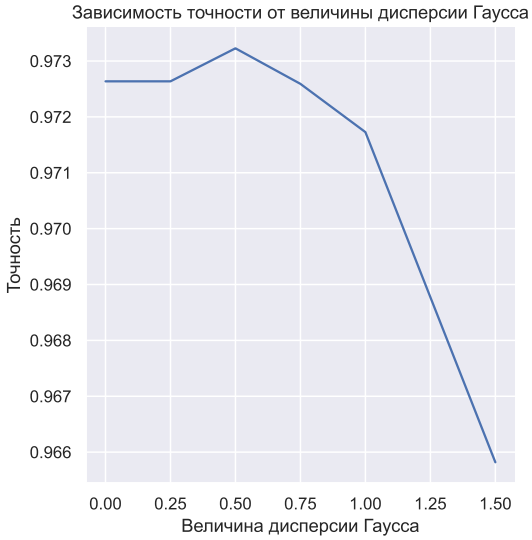
В цикле перебираются все возможные значения смещения по каждой из осей: -3, -2, -1, 0, 1, 2, 3. Формируется `shift_train` - выборка, каждый элемент которой получен из исходной выборки (состоит из 2000 элементов) применением функции `scipy.ndimage.shift`, сдвигающей изображение на определнное количество пикселей по каждой из осей. `aug_train_X` конкатенируется с `shift_train`, ответы, в свою очередь, конкатенируются с 2000 ответами, индексы которых были запомнены ранее. К расширенной выборке применяется кросс-валидация с 3 фолдами точности. Среднее по фолдам качество сохраняется для каждой комбинации смещений. В начале каждой итерации цикла



`aug_train_X` и `aug_train_y` остаются неизменными. Строится график зависимости точности от корня из суммы квадратов сдвигов по каждой из осей, т. е. от абсолютной величины смещения. По графику видно, что при любом смещении качество ухудшается. Это, возможно, происходит из-за того, что смещение вызывает изменение "центра тяжести"цифры, а у каждой он свой - и это отличает одну цифру от другой. Таким образом, смещение - плохое преобразование. Признаки не будут размножаться с помощью этого преобразования и `aug_train_X` и `aug_train_y` остаются неизменными.

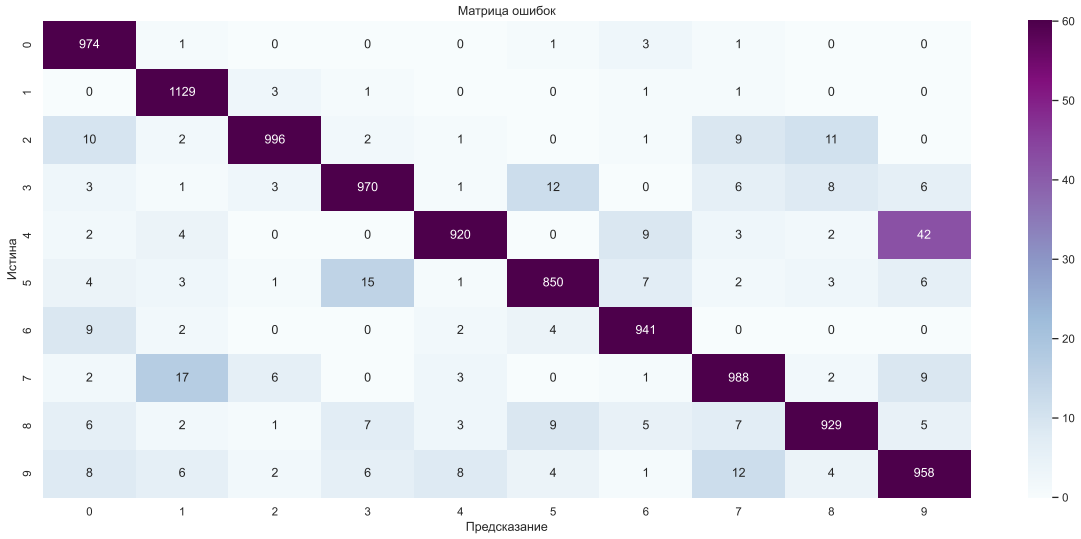
### 5.1.3 Фильтр Гаусса

Случайным образом выбираются 2000 элементов из `aug_train_X`, их индексы сохраняются. Эти 2000 объектов будут непосредственно преобразованы.




В цикле перебираются все возможные значения дисперсии фильтра Гаусса: 0, 0.25, 0.5, 0.75, 1, 1.5. Формируется `disp_train` - выборка, каждый элемент которой получен из исходной выборки (состоит из 2000 элементов) применением функции `skimage.filters.gaussian`, применяющей фильтр Гаусса с заданным значением дисперсии. `aug_train_X` конкатенируется с `disp_train`, ответы, в свою очередь, конкатенируются с 2000 ответами, индексы которых были запомнены ранее. К расширенной выборке применяется кросс-валидация с 3 фолдами точности. Среднее по фолдам качество сохраняется для каждого значения дисперсии. В начале каждой итерации цикла `aug_train_X` и `aug_train_y` остаются неизменными, конкатенация только для кросс-валидации. Строится график зависимости точности от величины дисперсии Гаусса. По графику видно, что наибольшее улучшение качества достигается для дисперсии 0.5. Таким образом, фильтр Гаусса - полезное преобразование.

С помощью библиотеки `sklearn` строится матрица ошибок на `aug_train_y`.



Заметно, что в результате фильтра цифра '1' стала чаще путаться с цифрой '7', '3' стала чаще путаться с цифрой '5', поскольку их размытые контуры похожи. Качество на тестовой выборке после аугментации: 0.9655. Качество ухудшилось на 0.0009. Но размытие - полезное преобразование, поскольку не позволяет сделать ошибку для цифр с различными очертаниями (после размытия от цифры остается только очертание).

Принимается решение добавить `disp_train` для наилучшей дисперсии в обучающую выборку.

### 5.1.4 Морфологические операции

Дилатация увеличивает толщину тонкий линий, эрозия нормализует толщину толстых линий, что полезно для изображений цифр жирным шрифтом, открытие убирает шумы с фона изображения, закрытие убирает шумы с самих линий цифр. Во всех преобразованиях ядро, скользящее по изображению, будет иметь размер в 4 пикселя.

К выборке `aug_train_X` применяется кросс-валидация с 3 фолдами точности для оценки качества до размножения выборки. Исходное качество: 0.97075. Эти данные сохраняются. Случайным образом выбираются 2000 элементов, их индексы сохраняются, поскольку далее преобразованным объектам нужно сопоставить корректные ответы. Эти 2000 объектов будут непосредственно преобразованы.

В цикле перебираются все рассматриваемые типы морфологических операций: эрозия, дилатация, открытие, закрытие. Формируются `erosion_train`, `dilation_train`, `opening_train`, `closing_train` - выборки, каждый элемент которых получен из исходной выборки (состоит из 2000 элементов) применением функций `cv.erode`, `cv.dilate`, `cv.morphologyEx`, совершающих морфологические преобразования. Выполняются конкатенации аналогично предыдущим этапам эксперимента. Значение качества на кросс-валидации:

- Открытие 0.9596
- Закрытие 0.9598
- Исходное 0.9708
- Эрозия 0.9714
- Дилатация 0.9718

Эрозия и дилатация дают небольшое улучшение точности. Открытие и закрытие ухудшает качество, поскольку все изображения довольно четкие и имеют фон без явных шумов, открытие и закрытие здесь излишни.

Принимается решение использовать дилатация в аугментацию всей исходной выборки.

## 5.2 Общий вывод

В ходе эксперимента установлено, что поворот на -5 градусов, фильтр Гаусса с дисперсией 0.5 и дилатация могут улучшать качество. Поэтому нужно размножить исходную обучающую выборку `train_X` из 60000 элементов с помощью этих преобразований. Исходное качество на кросс-валидации: 0.9754, на тесте: 0.9752. С помощью каждого преобразования добавляются 2000 элементов и ответов на них, аналогично тому, как это делалось ранее. Итоговое качество на кросс-валидации: 0.9779, на тесте: 0.9757. Таким образом, такая аугментация действительно улучшила точность.

# 6 Эксперимент №6

## 6.1 Постановка

Необходимо провести аугментацию тестовой выборки для тех же преобразований, что и в 5 эксперименте. Обучение происходит на оригинальном датасете.

Создаётся объект класса классификатора. Производится его обучение на обучающей выборке. Для каждого преобразования формируется пустой список предсказаний и запускается цикл для всевозможных значений параметров. В теле цикла для каждого элемента тестовой выборки производится соответствующее преобразование с соответсвующим параметром. Так формируется новая тестовая выборка. Именно на ней производится предсказание. Предсказание добавляется в список пресказаний. Список предсказаний имеет вид списка из списков: каждый элемент-список - это предсказания для конкретного объекта для разных значений параметров. Итоговое предсказание для каждого списка предсказаний формируется следующим образом: если более половины предсказаний совпадают с истиной, то предсказание верное, иначе - нет. На этом принципе считается качество.

Но для построения матрицы ошибок нужно иметь массив пресказаний, а не просто факт того, верное оно или нет. Может произойти так, что никакое предсказание не встречается более половины раз от общего количества предсказаний для каждого объекта выборки. Поэтому каждый элемент массива предсказаний для всей выборки - это наиболее частое предсказание для каждого вектора предсказаний. Из вектора предсказаний и вектора истинных значений строится матрица ошибок.

### 6.2 Результаты и выводы

- Точность для поворотов: 0.9666
- Точность для сдвигов: 0.9959
- Точность для фильтра Гаусса: 0.9684
- Точность для морфологических операций: 0.8806

Точность для сдвигов значительно увеличилась. Это связано с тем, количество различных параметров сдвига велико - 49. Изображение при сдвигах меняется незначительно, значит наиболее вероятно, что более половины голосов будет за один и тот же вариант, который является истинным значением. Наибольшее количество ошибок при сдвигах пришлось на пары цифр ’9’ и ’4’, ’9’ и ’7’. Эти пары очень похожи начертанием, и их "центры тяжести"очень близки друг к другу. Именно поэтому даже небольшие сдвиги координально меняли предсказание.

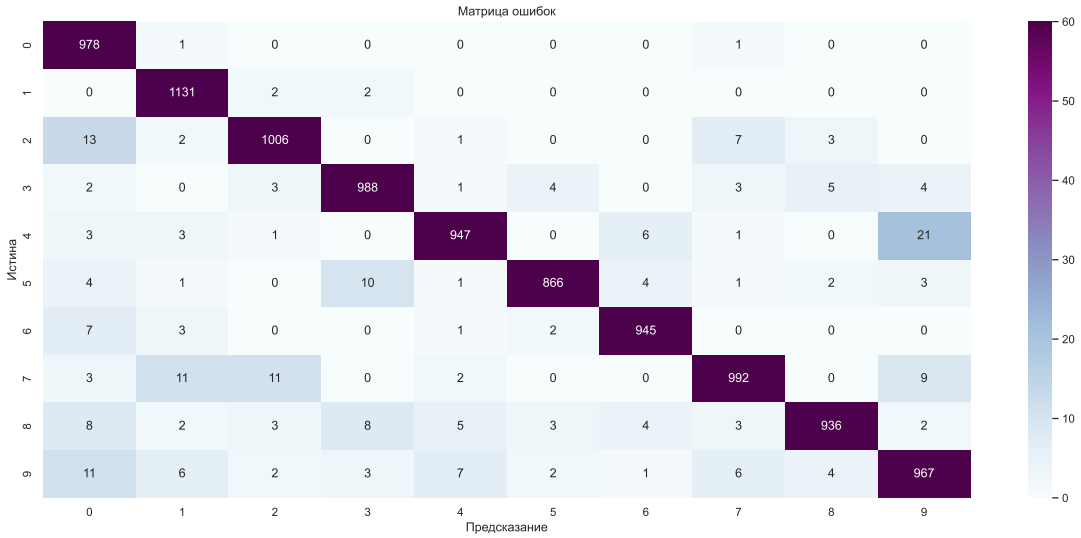
Поворотами очень похожими можно сделать следующие пары: ’4’ и ’9’, ’1’ и ’7’, обогащение тестовой выборки таким преобразованием снизило точность.

Фильтр Гаусса практически не влияет на точность, в то время как морфологические операции существенно её снизили. Таким образом, большинство преобразований тестовой выборки не улучшили точность, а даже снизили её. Это может быть связано с малым количеством объектов в тестовой выборке, отсутствием факта взвешенности (только голосование, наиболее близкие объекты не имеют преимуществ перед более далёкими) и малым количеством различных значений параметра. Ниже приведены матрицы ошибок для поворотов и сдвигов соответственно.

## 7 Заключение

Отчёт посвящён метрическим алгоритмам классификации и методам работы с изображениями. Результаты экспериментов позволили сделать следующие ключевые выводы:

- `brute` - наиболее быстрая стратегия поиска ближайших соседей
- Косинусная метрика и взвешенный метод 4 ближайших соседей позволяют наиболее точно распознавать рукописные цифры методом KNN
- С помощью сдвигов тестовой выборки можно достигнуть очень высокую точность: 0.9959


Матрица ошибок
Истина
Предсказание
0 978 1 0 0 0 0 0 1 0 0
1 0 1131 2 2 0 0 0 0 0 0
2 13 2 1006 0 1 0 0 7 3 0
3 2 0 3 988 1 4 0 3 5 4
4 3 3 1 0 947 0 6 1 0 21
5 4 1 0 10 1 866 4 1 2 3
6 7 3 0 0 1 2 945 0 0 0
7 3 11 11 0 2 0 0 992 0 9
8 8 2 3 8 5 3 4 3 936 2
9 11 6 2 3 7 2 1 6 4 967
0 1 2 3 4 5 6 7 8 9
60 50 40 30 20 10 0


Матрица ошибок
Истина
Предсказание
0 971 1 1 0 0 0 4 1 2 0
1 0 1129 4 0 0 0 1 0 0 1
2 9 1 1009 1 0 0 1 11 0 0
3 1 0 11 975 0 5 0 12 5 1
4 2 4 0 0 956 0 8 2 2 8
5 4 0 3 18 1 845 15 1 3 2
6 7 3 0 0 1 1 946 0 0 0
7 1 13 3 0 1 0 0 1006 0 4
8 2 2 7 12 2 4 10 13 919 3
9 12 11 8 10 33 4 1 52 3 875
0 1 2 3 4 5 6 7 8 9
60 50 40 30 20 10 0