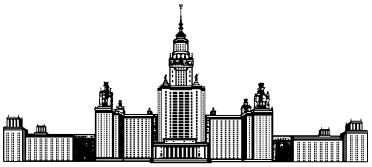
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

ОТЧЁТ ПО ЗАДАНИЮ №1

«Метрические алгоритмы классификации»

Выполнил:
студент 3 курса 317 группы
Суглобов Кирилл Алексеевич

Москва, 2021

Содержание

Введение

В задании требовалось написать реализации метода ближайших соседей и кросс-валидации, ознакомиться с метрическими алгоритмами классификации и методами работы с изображениями. В отчёте описывается ход экспериментов, которые выполнялись с помощью написанных на Python модулей, на датасете MNIST, и их результаты. Далее, если не оговорено иного, предполагается следующее разбиение датасета:

- Обучающая выборка: первые 60 000 объектов
- Тестовая выборка: последние 10 000 объектов

Эксперимент №1: самый быстрый алгоритм

В эксперименте №1 требуется сравнить время работы алгоритмов написанного kNN-классификатора на примере поиска 5-ти ближайших соседей для каждого объекта тестовой выборки в зависимости от количества признаков (от размерности признакового пространства). Тестовая выборка берётся блоками по 1000 объектов: это используется только в алгоритмах, которые строят в памяти матрицу попарных расстояний (переборные алгоритмы `my_own`, `brute`).

На рис. 1 изображён график для малого числа признаков. Видно, что `tree`-стратегии (деревья) имеют преимущество только в случае малого числа признаков: здесь до 10 признаков преимущество у `kd_tree` и `ball_tree`, до 20 признаков преимущество сохраняется для `kd_tree`. На малом числе признаков `ball_tree` медленнее `kd_tree`. После примерно 20-ти (22-х) признаков преимущество по скорости работы берёт на себя `brute`-стратегия. На время работы переборных алгоритмов количество признаков почти не влияет. Теперь рассмотрим ситуацию, когда > 100 признаков.

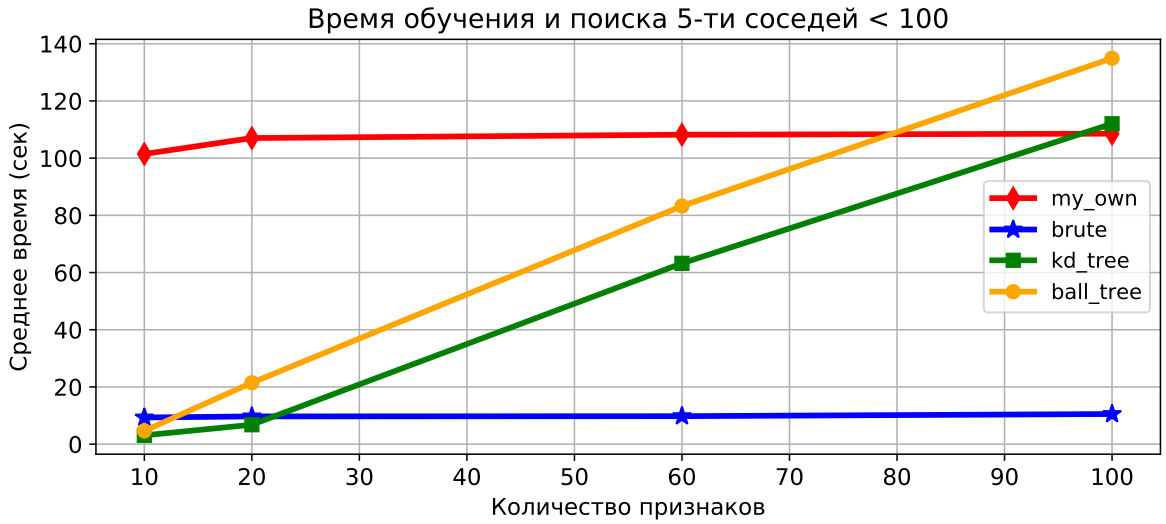


Рис. 1

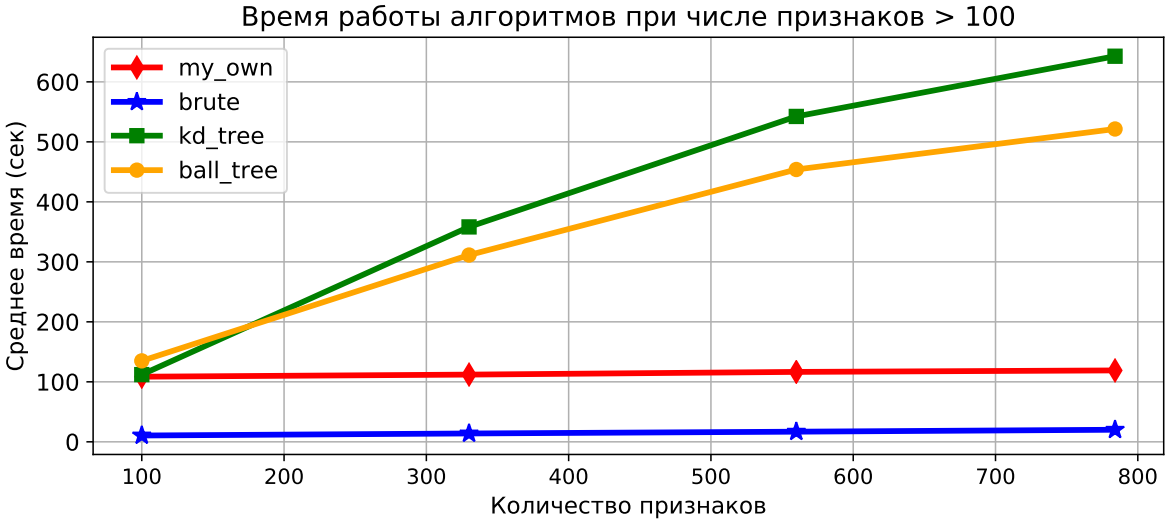


Рис. 2

На рис. 2 изображён график для большого числа признаков (> 100). Видно, что время работы `tree`-стратегий сильно быстро растёт с ростом количества признаков. Это идёт вразрез с логарифмической асимптотикой работы этих стратегий, которая верна для случаев сбалансированных деревьев с «хорошим» (равномерным) распределением признаков в признаковом пространстве. Но здесь размерность признакового пространства растёт, что вызывает «проклятие размерности»: 60 000 объектов с 784-мя признаками у каждого недостаточно для равномерного покрытия этого пространства.

Рост количества признаков до 784 по-прежнему, как и в случае малого количества признаков, почти не влияет на время работы переборных алгоритмов: `brute` и `my_own`.

Далее будет использоваться алгоритм `brute`, как самый быстрый, и алгоритм `my_own`, по необходимости. `tree`-стратегии показали, что они неэффективны при размерностях нашей задачи.

Эксперименты №2 и №3: выбор лучшей метрики, добавление весов

В эксперименте №2 требуется оценить по кросс-валидации с 3 фолдами качество, т.е *accuracy* (долю правильно предсказанных ответов) и время работы алгоритма поиска k ближайших соседей в зависимости от следующих факторов:

(a) k от 1 до 10 (только влияние на качество).

(b) Используется евклидова или косинусная метрика.

В эксперименте №3 требуется сравнить взвешенный метод k ближайших соседей с методом без весов при тех же фолдах и параметрах. Во взвешенном методе голос объекта равен не $w = 1$ (случай без весов), а $w = \frac{1}{d+10^{-5}}$, где d — расстояние (по одной из метрик в признаковом пространстве) между объектом и рассматриваемым соседом.

Кроме того, требуется измерить время на кросс-валидации и подобрать оптимальное для качества значение k — количество ближайших соседей.

В эксперименте использовался алгоритм `brute` с метриками *euclidean* и *cosine*, с весами и без весов.

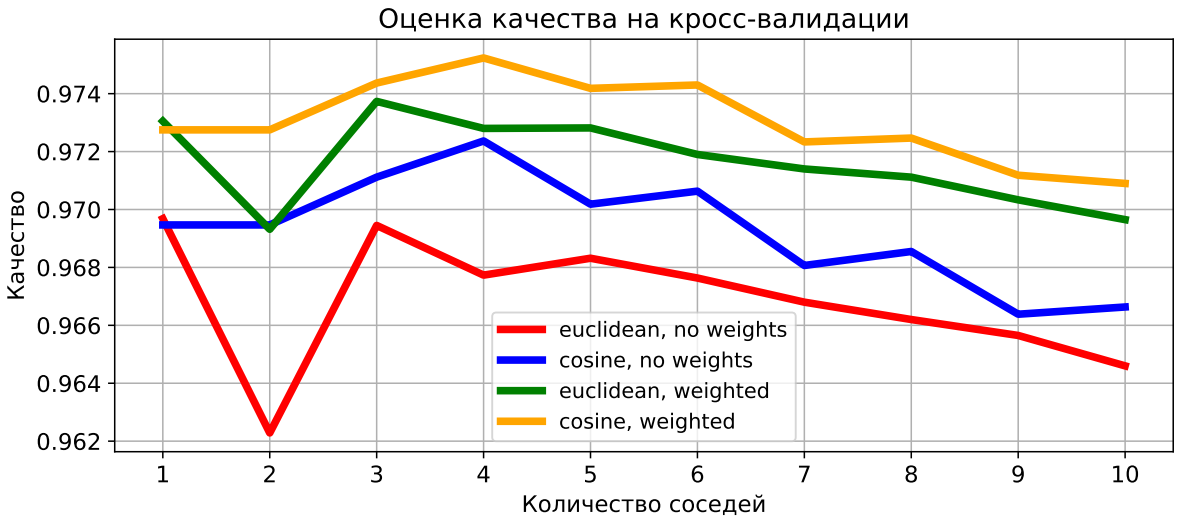


Рис. 3

По кросс-валидации видно, что косинусная метрика даёт большее качество модели. Использование весов улучшает качество для обеих метрик.

Также по графику рис. 3 видно, что оптимально использование метода 4-х ближайших соседей (чётность не влияет, потому что веса $w \neq 1$)

Напомним, что качество на кросс-валидации усреднялось по 3-м фолдам. Лучшее среднее качество: **97.52(3)%**,при методе $k = 4$ ближайших соседей с весами и косинусной метрикой.

В собственном переборном алгоритме `my_own` эффективно реализован подсчёт расстояний до соседей: один раз строится таблица с расстояниями до самого дальнего соседа, потом «отбрасываются лишние соседи».

Также требовалось измерить время поиска $k = 10$ соседей в процессе кросс-валидации. Результат на рис. 4.

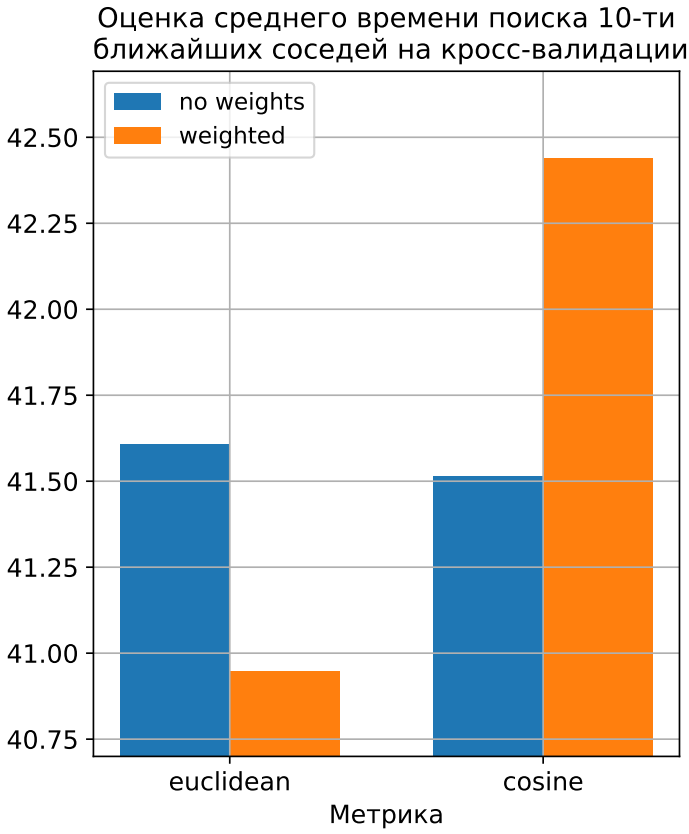


Рис. 4

При подсчёте расстояния по метрике *euclidean* использовалась формула:

$$\rho_{L_2}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle} = \sqrt{\|\mathbf{x}\|_2^2 - 2\langle \mathbf{x}, \mathbf{y} \rangle + \|\mathbf{y}\|_2^2} \tag{1}$$

При подсчёте расстояния по метрике *cosine* использовалась формула:

$$\rho_{cos}(\mathbf{x}, \mathbf{y}) = 1 - \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = 1 - \left\langle \frac{\mathbf{x}}{\|\mathbf{x}\|}, \frac{\mathbf{y}}{\|\mathbf{y}\|} \right\rangle \tag{2}$$

Эксперимент №4: качество модели

В эксперименте №4 требуется оценить качество лучшей модели, сравнить с качеством на кросс-валидации с 3-мя фолдами, сравнить с качеством в Интернете и выяснить, на каких объектах модель ошибается.

На экспериментах №1-3 были подобраны лучшие гиперпараметры и алгоритмы — взвешенные переборные алгоритмы с метрикой *cosine* и $k = 4$: `brute` (самый быстрый) и `my_own`. Обучим модель, соответствующую алгоритму `brute`, на обучающей выборке, получим предсказания на тестовой выборке, проанализируем их.

- Качество на тестовой выборке: **97.52%**
- Среднее качество на кросс-валидации: **97.52(3)%**

Различие незначительное, кросс-валидация сработала хорошо, в ней применялся следующий алгоритм: сначала датасет MNIST (обучающая выборка) перемешивался, а потом последовательно разбивался на фолды.

Качество лучших алгоритмов на датасете MNIST, согласно рейтингу [1]: **99.87%**, результат 2020-го года. В своей статье [2] авторы используют Homogeneous Vector Capsules и свёрточные нейронные сети.

Вернёмся к нашей модели, посмотрим, где она ошибается. На рис. 5 изображена матрица ошибок лучшего алгоритма.

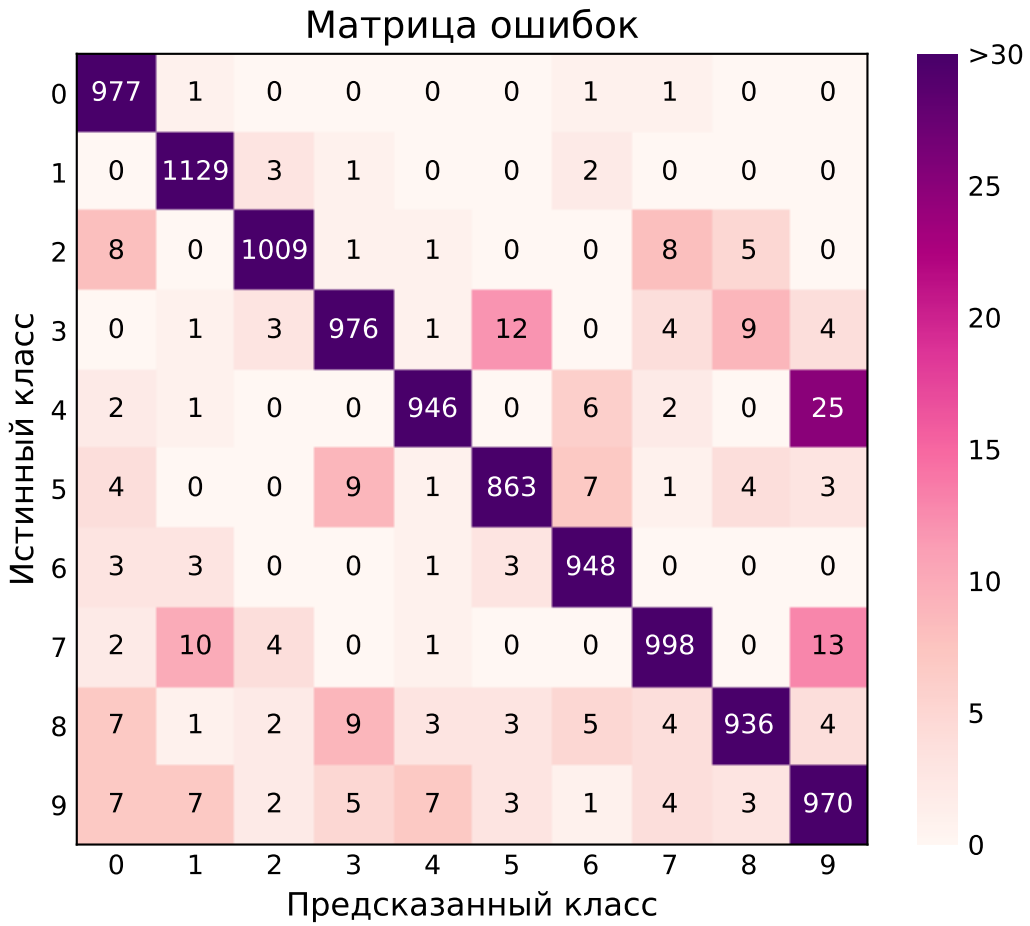


Рис. 5

Визуальный анализ матрицы ошибок (рис. 5) показал, что:

- Цифры ***0*** и ***1*** классифицируются лучше всего (меньше всего ошибок). Видимо, это из-за небольшого количества «простых линий» в их символах.
- Цифры ***8*** и ***9*** классифицируются хуже всего (больше всего ошибок). Видимо, это из-за «переплетений линий» в их символах.

- Алгоритм нередко распознаёт истинную ***3*** как ***5*** (12 раз) и наоборот (9 раз) (иногда ошибки симметричны, но в основном — нет)
- Самые частые ошибки:
  1. (25 раз) Распознавание истинной ***4*** как ***9***
  2. (13 раз) Распознавание истинной ***7*** как ***9***
  3. (12 раз) Распознавание истинной ***3*** как ***5***
  4. (10 раз) Распознавание истинной ***7*** как ***1***

Далее на рис. 6, рис. 7, рис. 8, рис. 9 визуализированы некоторые группы ошибочно классифицированных объектов с общими чертами.

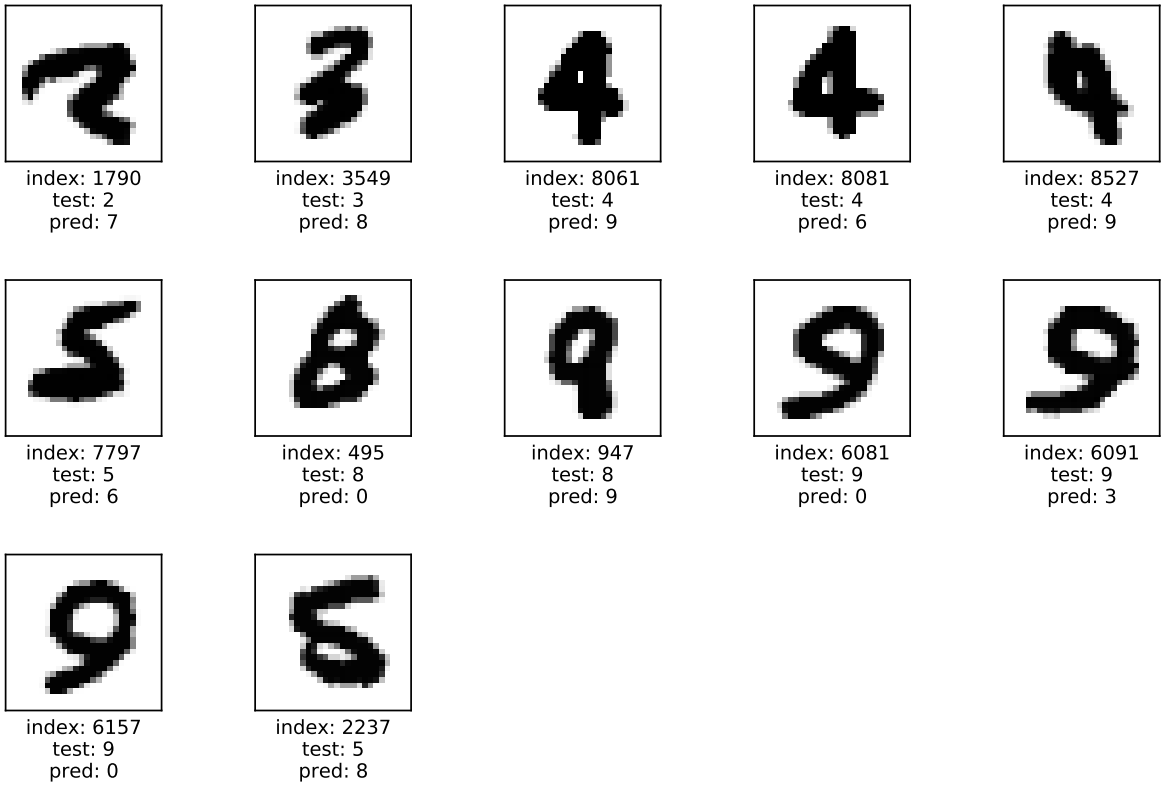


Рис. 6: Группа 1: слишком жирное написание

Эксперимент №5: аугментация обучающей выборки

В эксперименте №5 требуется размножить (аугментировать) обучающую выборку с помощью следующих преобразований:

- Поворот изображений на: $\pm 5^\circ$, $\pm 10^\circ$, $\pm 15^\circ$
- Сдвиг изображений на: ± 1px, ± 2px, ± 3px (по каждой из двух размерностей)
- Применение к изображениям фильтра Гаусса с дисперсией: 0.5, 1.0, 1.5

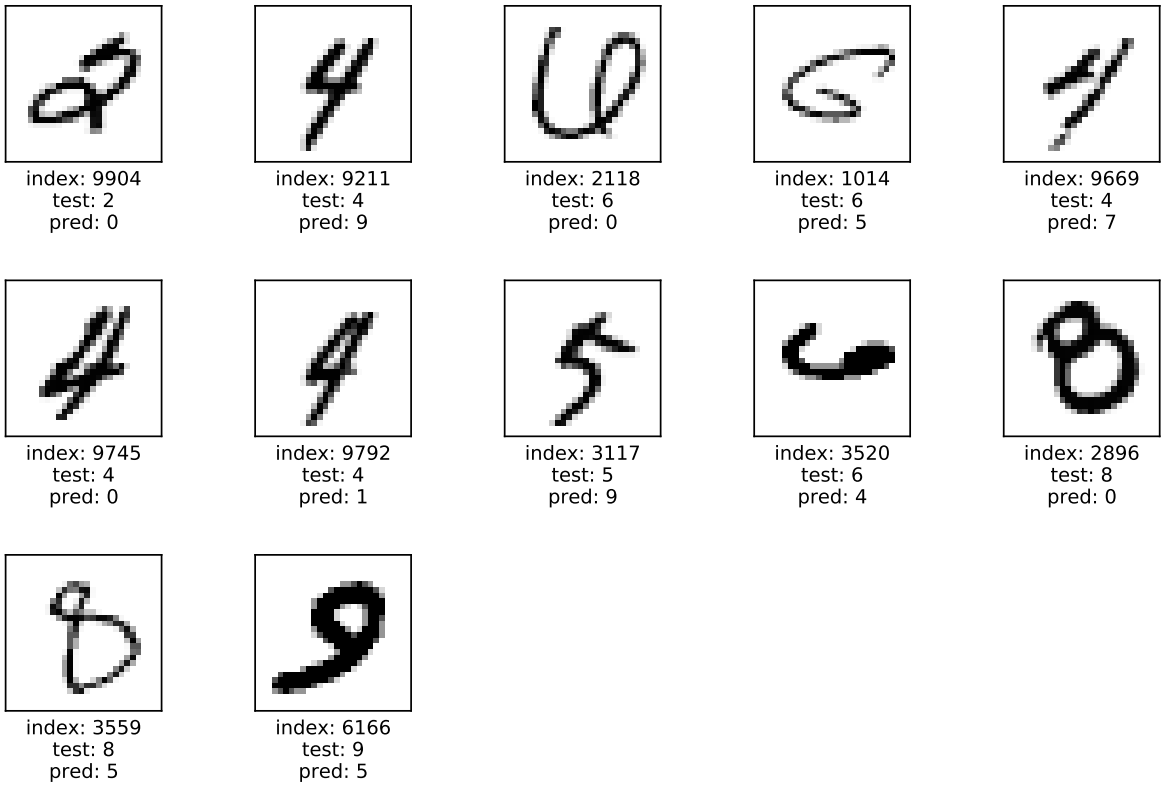


Рис. 7: Группа 2: повёрнутые цифры

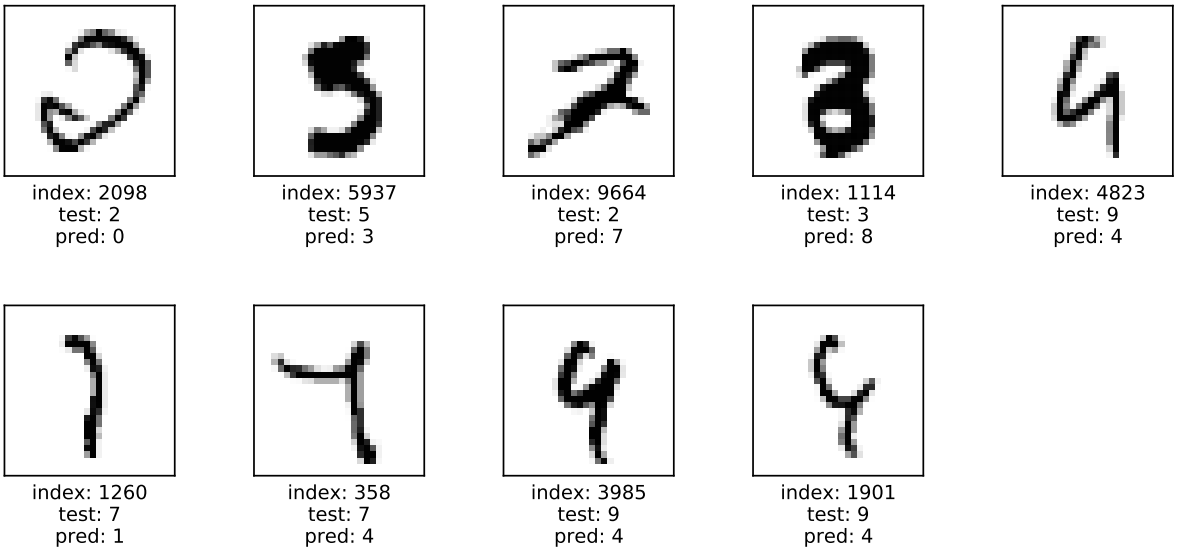


Рис. 8: Группа 3: цифры, очень похожие на другие (неоднозначная классификация)

После чего требуется определить качество модели на различных преобразованиях и влияние этих преобразований.

Выполним кросс-валидацию на лучшем (то есть метрика *cosine*, алгоритм с весами) алгоритме, `brute`-алгоритме. Выясним оптимальное число k ближайших соседей при разных трансформациях. Выборка преобразуется таким образом: к исходной выборке прибавляется трансформированная. Например, в случае сдвига

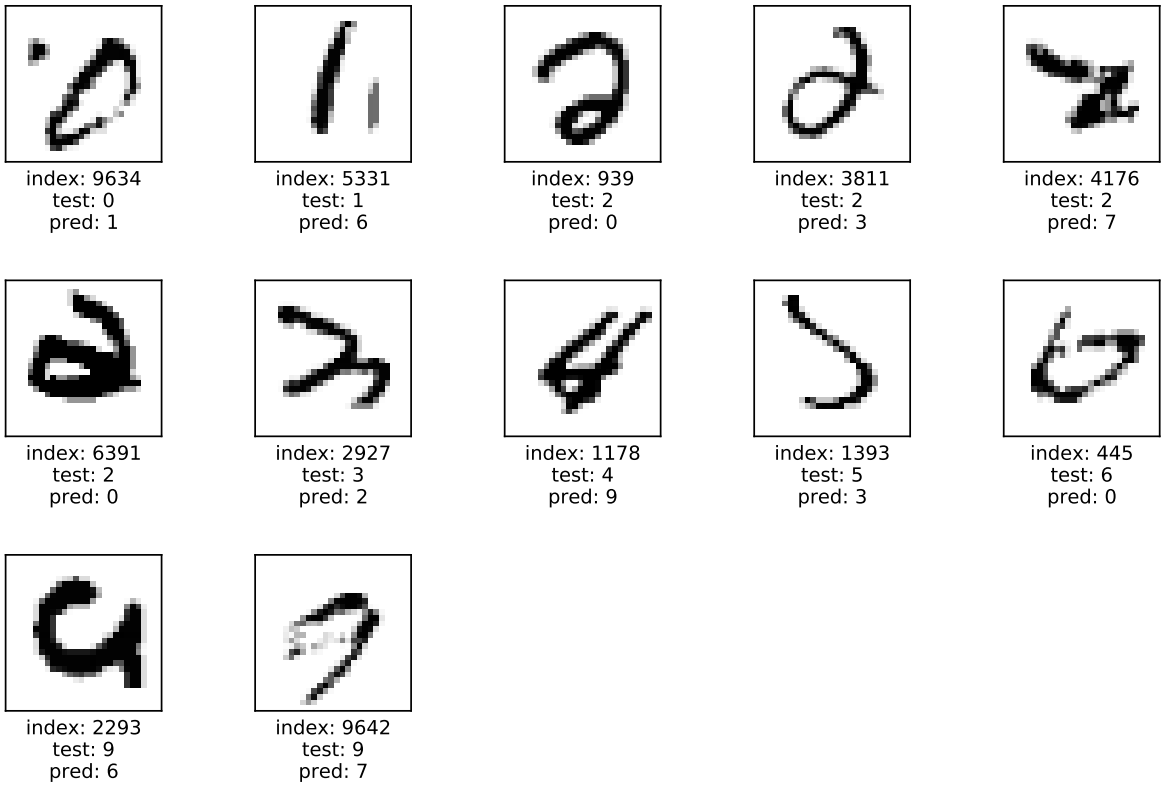


Рис. 9: Группа 4: трудно классифицируемые символы

аугментированная обучающая выборка будет состоять из исходной выборки и двух трансформированных (преобразование по одной из осей и в противоположном направлении, например +1px и -1px по горизонтали). То есть в этом примере аугментированная выборка будет в три раза больше исходной обучающей выборки.

Далее на рис. 10 приведены результаты кросс-валидации. Для сравнения на графике пунктиром приводится график этого же лучшего алгоритма, с теми же гиперпараметрами, но без аугментации.

Исходя из графиков, лучшие параметры трансформаций:

- Поворот изображений на: $\pm 5^\circ$
- Сдвиг изображений на: $\pm$1px (по каждой из двух размерностей)
- Применение к изображениям фильтра Гаусса с дисперсией: 0.5

Далее в этом эксперименте мы объединим сдвиги по разным осям (при трансформации сдвигом, выборка увеличится в 5 раз: исходная выборка + по 2 трансформированные выборки по каждой из осей)

Также, исходя из графиков, оптимальный параметр k:

- При повороте: $k = 2$
- При сдвиге: $k = 4$
- При размытии: $k = 6$

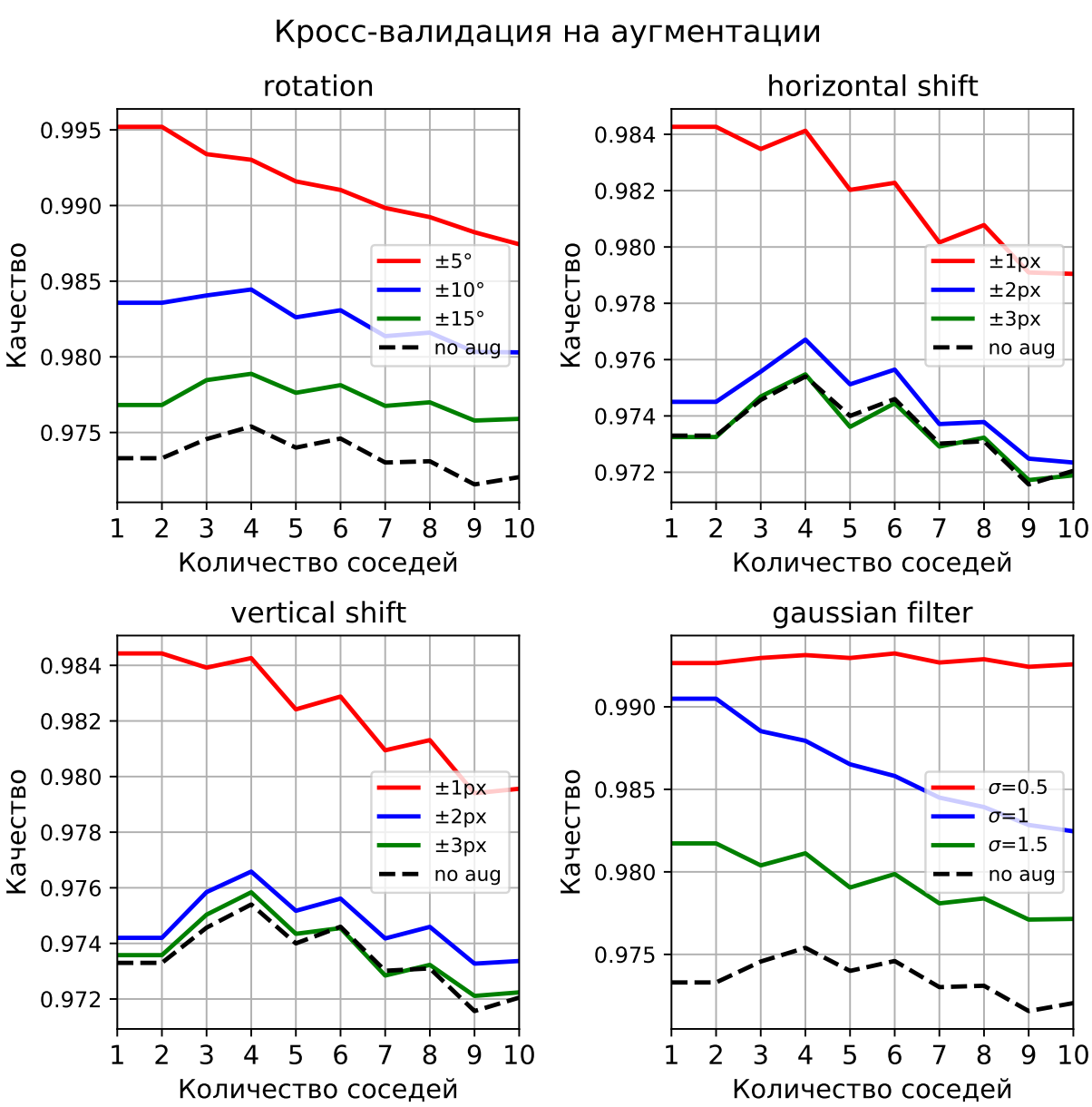


Рис. 10

Руководствуясь этими гиперпараметрами, сделаем предсказания на всех видах аугментированных по лучшим параметрам трансформаций выборках. То есть предсказания на выборках, аугментированных с помощью поворота на $\pm 5°$, сдвига на ± 1px, размытия фильтром Гаусса с дисперсией 0.5, и на комбинациях этих аугментированных обучающих выборок. Назовём эти варианты аугментации и их комбинации как: **rotation**, **shift**, **gauss**, **rotation+shift**, **rotation+gauss**, **shift+gauss**, **rotation+shift+gauss**. Вариант работы алгоритма без аугментации обучающей выборки (нужно для сравнения) назовём **no_aug**.

Далее на рис. 11 и на рис. 12 представлены матрицы ошибок для всех описанных случаев. Видно, как разные аугментации уменьшают число различных ошибок в матрице.

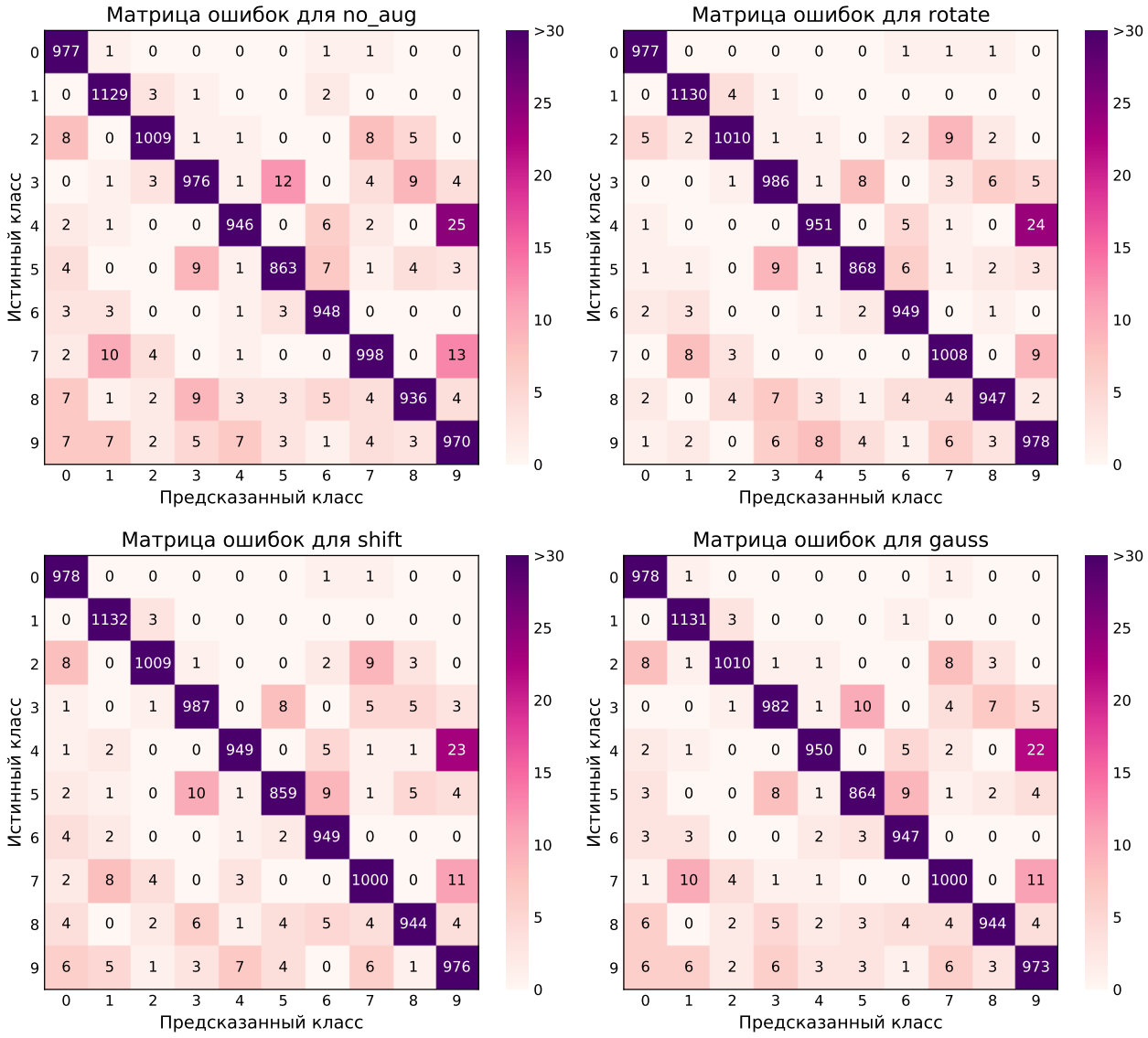


Рис. 11: Матрицы ошибок на аугментированных выборках

Преобразования поворота исправляли ошибки классификации на повёрнутых цифрах, преобразования смещения — на нецентрированных цифрах, а преобразование размытия фильтром Гаусса — на жирно написанных цифрах.

По матрицам ошибок явно видно, как уменьшаются ошибки на различных преобразованиях. Две самые распространённые ошибки: распознавания истинной ***4*** как ***9*** и распознавания истинной ***7*** как ***9*** — эти ошибки были уменьшены на комбинациях, в которых есть поворот: **rotate+shift**, **rotate+shift+gauss**, **rotate+shift**, **rotate+gauss**. Также примерно все методы уменьшили распознавание истинного ***0*** как другой цифры. А вот количество ошибок распознавания истинной ***2*** как ***7*** увеличилось или осталось прежним.

На табл. 1 приведён итог эксперимента №5. Лучшее качество **97.35%** достигается при сочетании поворотов на $\pm 5^\circ$ и сдвигов на ± 1px (по обеим осям).

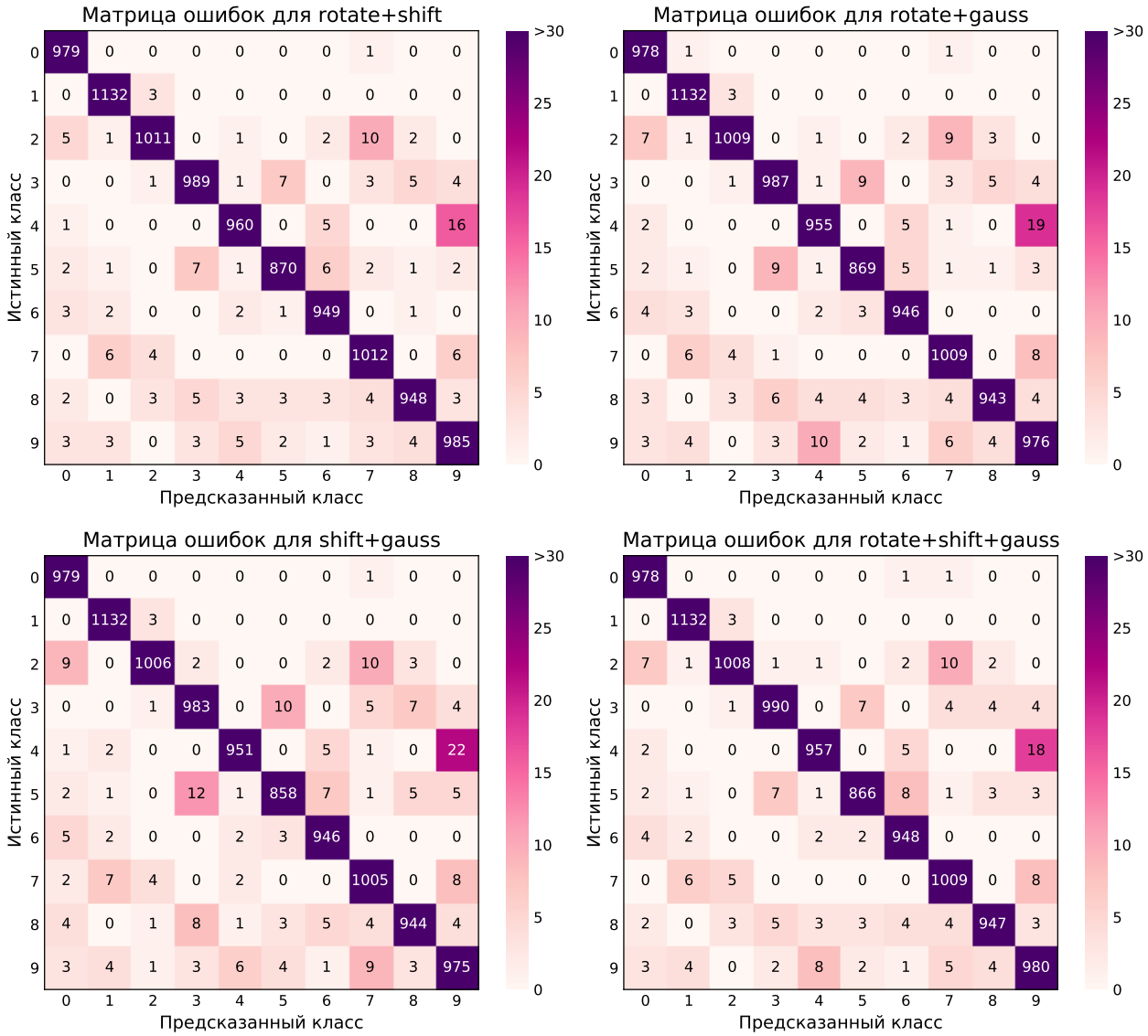


Рис. 12: Матрицы ошибок на комбинациях аугментированных выборок

Аугментированная выборка	Качество (%)
no_aug	97.52
rotate	98.04
shift	97.83
gauss	97.78(9)
rotate+shift	98.35
rotate+gauss	98.04
shift+gauss	97.78(9)
rotate+shift+gauss	98.15

Таблица 1: Качество модели на аугментированных обучающих выборках при всех типах преобразований и при всех комбинациях. Лучшее качество: **98.35%**.

Вывод из эксперимента №5: с помощью различных аугментаций на обучающей

выборке можно исправлять качественно разные ошибки алгоритма, корректировать его.

Эксперимент №6: аугментация тестовой выборки

В эксперименте №6 требуется размножить объекты тестовой выборки, то есть сделать аугментацию тестовой выборки, а для её преобразований использовать те же, что и в эксперименте №5:

- Поворот изображений на: $\pm 5^\circ$, $\pm 10^\circ$, $\pm 15^\circ$
- Сдвиг изображений на: $\pm$1px, $\pm$2px, $\pm$3px (по каждой из двух размерностей)
- Применение к изображениям фильтра Гаусса с дисперсией: 0.5, 1.0, 1.5

После чего требуется определить качество модели на различных преобразованиях и влияние этих преобразований, аналогично эксперименту №5, но только аугментация не на обучающей, а на тестовой выборке. Будем искать расстояния при различных преобразованиях тестовой выборки по стратегии наименьшего расстояния, что согласуется с гипотезой компактности [3] и идей метода k ближайших соседей. То есть среди всех аугментированных тестовых выборок выбирается наименьшее расстояние для этого объекта.

Выберем параметры трансформации по той же схеме, что и в эксперименте №5.

Выполним кросс-валидацию на лучшем (то есть метрика *cosine*, алгоритм с весами) алгоритме, `my_own`-алгоритме. Выясним оптимальное число k ближайших соседей при разных трансформациях.

Далее на рис. 13 приведены результаты кросс-валидации. Для сравнения на графике пунктиром приводится график этого же лучшего алгоритма, с теми же гиперпараметрами, но без аугментации.

Исходя из графиков, лучшие параметры трансформаций остались теми же, только фильтр Гаусса в этот раз ухудшал качество, поэтому для простоты оставим его прежним, с дисперсией 0.5:

- Поворот изображений на: $\pm 5^\circ$
- Сдвиг изображений на: $\pm$1px (по каждой из двух размерностей)
- Применение к изображениям фильтра Гаусса с дисперсией: 0.5

Объединим сдвиги по разным осям.

Далее, как видно по графикам, оптимальный параметр k изменился по сравнению с предыдущим экспериментом. Теперь это везде $k = 4$:

- При повороте: $k = 4$
- При сдвиге: $k = 4$
- При размытии: $k = 4$

Аналогично предыдущему эксперименту, руководствуясь подобранными гиперпараметрами, сделаем предсказания на всех видах аугментированных по лучшим параметрам трансформаций выборках.

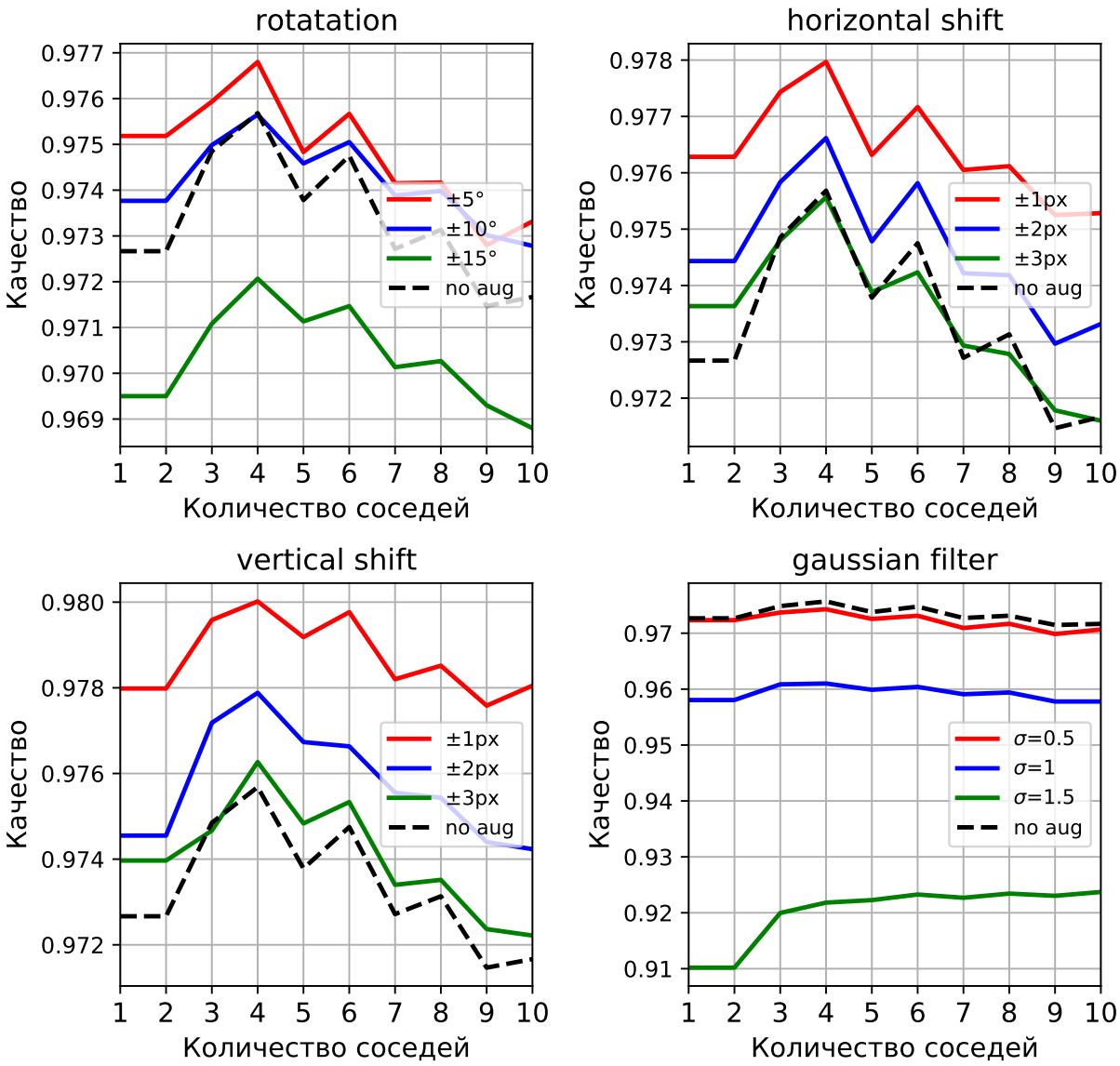


Рис. 13

Далее на рис. 14 и на рис. 15 представлены матрицы ошибок для всех описанных случаев.

Как и в прошлом эксперименте, преобразования поворота исправляли ошибки классификации на повёрнутых цифрах. Фильтр Гаусса справился с хуже, чем в эксперименте №5. В целом видно, что качество стало хуже в сравнении с предыдущим экспериментом.

По матрицам ошибок видно, что фильтр Гаусса ухудшает качество модели, в сравнении с экспериментом №5. Если в аугментации обучающей выборки фильтр Гаусса, «сглаживая» шум, обучает алгоритм лучше, то в аугментации тестовой выборки фильтр Гаусса «затирает» нужную для классификации информацию, внося в классификацию больше «неопределённости». Повороты и сдвиги вносят примерно равный полезный вклад, что в этом эксперименте, что в эксперименте №5. Но в случае аугментации обучающей выборки лучше работают повороты, а в случае

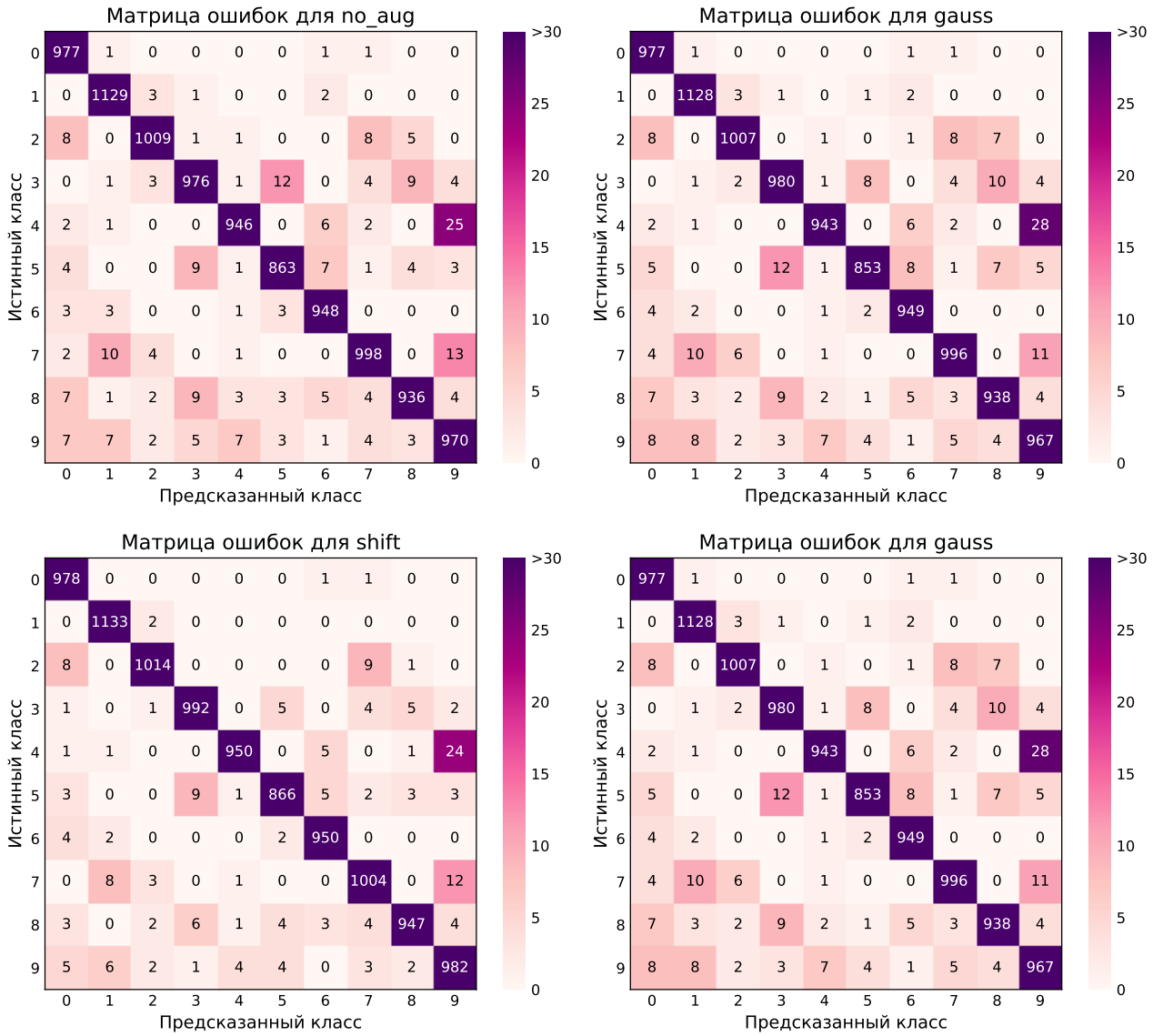


Рис. 14: Матрицы ошибок на аугментированных тестовых выборках

аугментации тестовой выборки — сдвиги.

На табл. 2 приведён итог эксперимента №6. Лучшее качество **97.35%** достигается сдвигах на ± 1px (по обеим осям).

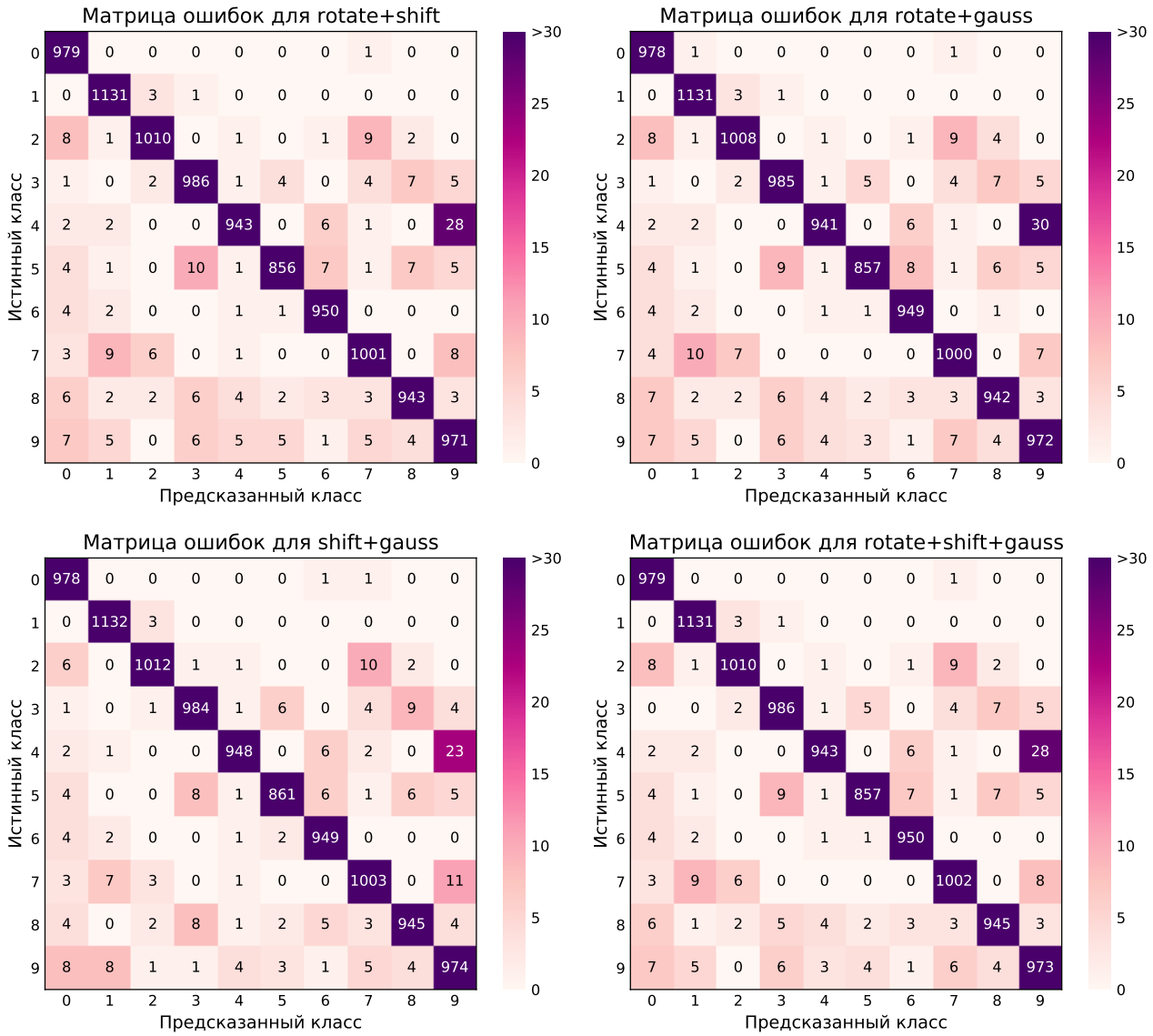


Рис. 15: Матрицы ошибок на комбинациях аугментированных тестовых выборок

Аугментированная выборка	Качество (%)
no_aug	97.52
rotate	97.6
shift	98.16
gauss	97.38
rotate+shift	97.7
rotate+gauss	97.63
shift+gauss	97.86
rotate+shift+gauss	97.76

Таблица 2: Качество модели на аугментированных тестовых выборках при всех типах преобразований и при всех комбинациях. Лучшее качество: **98.16%**.

Заключение

Итого на примере датасета MNIST был проведен ряд экспериментов с методом классификации *kNN*. В данном примере лучшие гиперпараметры оказались такими: метрика — косинусная, веса в алгоритме — да, $k = 4$. При этом, для нашей задачи эффективны (по скорости работы) переборные алгоритмы *kNN*, а не деревья. Сравнивались два различных метода аугментации выборок: обучающей и тестовой. Оба подхода имеют свои преимущества и недостатки, но лучший результат показала аугментация обучающей выборки: на ней лучшее качество **98.35%** достигается при сочетании поворотов и сдвигов. Итоговое качество — **98.35%**.

References

[1] *Image Classification on MNIST*. URL: https://paperswithcode.com/sota/image-classification-on-mnist. accessed: 14 October 2021.

[2] Adam Byerly, Tatiana Kalganova, and Ian Dear. *No Routing Needed Between Capsules*. 2020. arXiv: 2001.09136 [cs.CV].

[3] Nikolay Zagoruiko. *Measure of Similarity and Compactness in Competitive Space*. URL: http://math.nsc.ru/~wwwzag/5.%20FRiS--InfComp.pdf.