Московский Авиационный Институт

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №1 по курсу «Искусственный Интеллект»

Студент: В. П. Будникова

Преподаватель:

Группа: М80-307Б-19

Дата:

Оценка:

Подпись:

Москва, 2022

Задание:

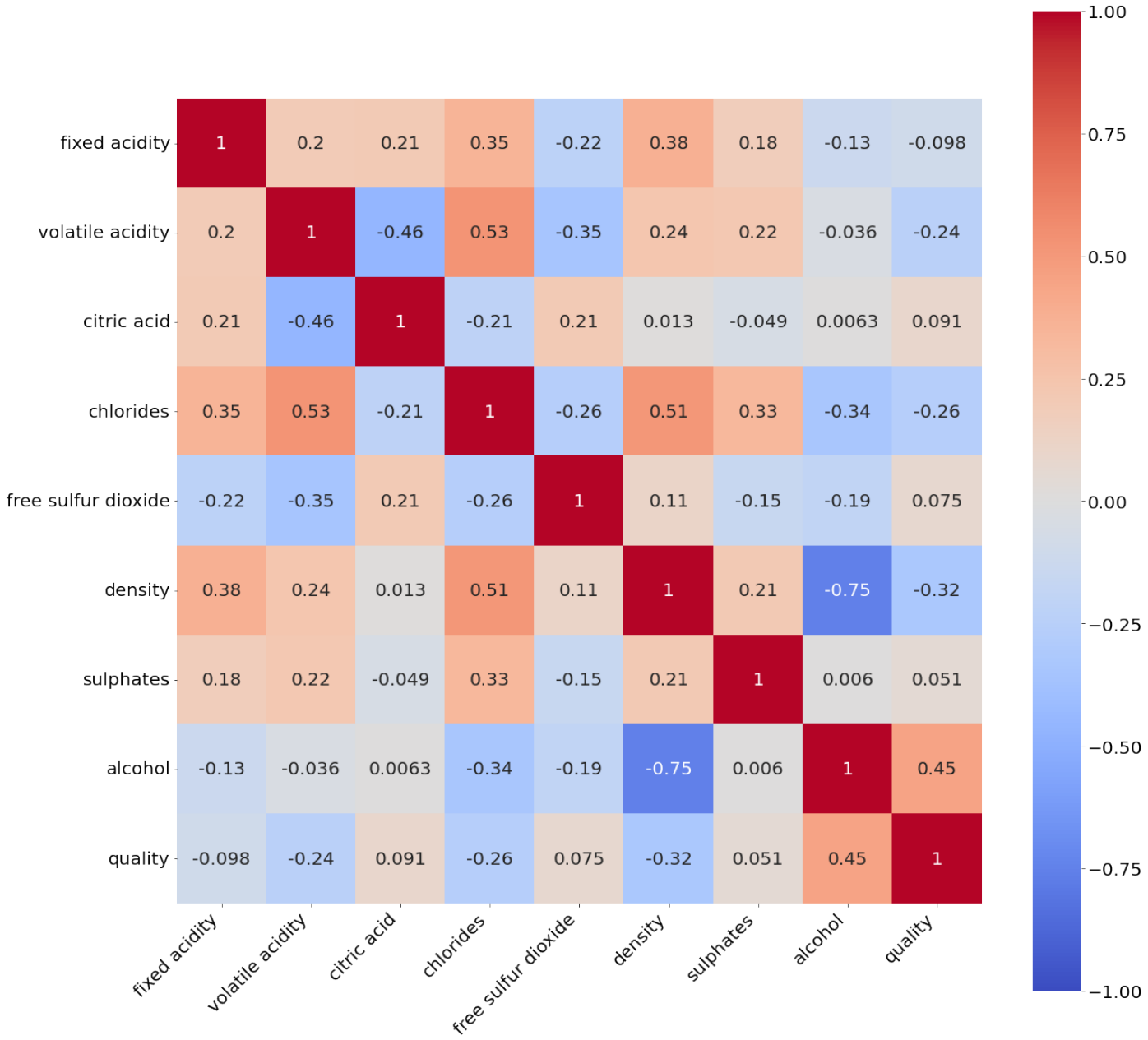
Реализовать следующие алгоритмы машинного обучения: Linear/ Logistic Regression, SVM, KNN, Naive Bayes в отдельных классах. Данные классы должны наследоваться от BaseEstimator и ClassifierMixin, иметь методы fit и predict. Организовать весь процесс предобработки, обучения и тестирования с помощью Pipeline. Проделать аналогично с коробочными решениями

Оборудование:

Ноутбук, процессор: 1,6 GHz 2-ядерный процессор Intel Core i5, память 8ГБ.

Ход работы:

Для начала, используя выводы анализа данных, удалим выбросы и элементы, обладающие слабой корреляцией с результатом.
Матрица корреляций:



Посмотрим, как данные распределяются по классам:

```
3 13
4 176
5 1895
6 2556
7 1007
8 180
9 5
```

Так как данные несбалансированные, разделим их на три более сбалансированных класса:

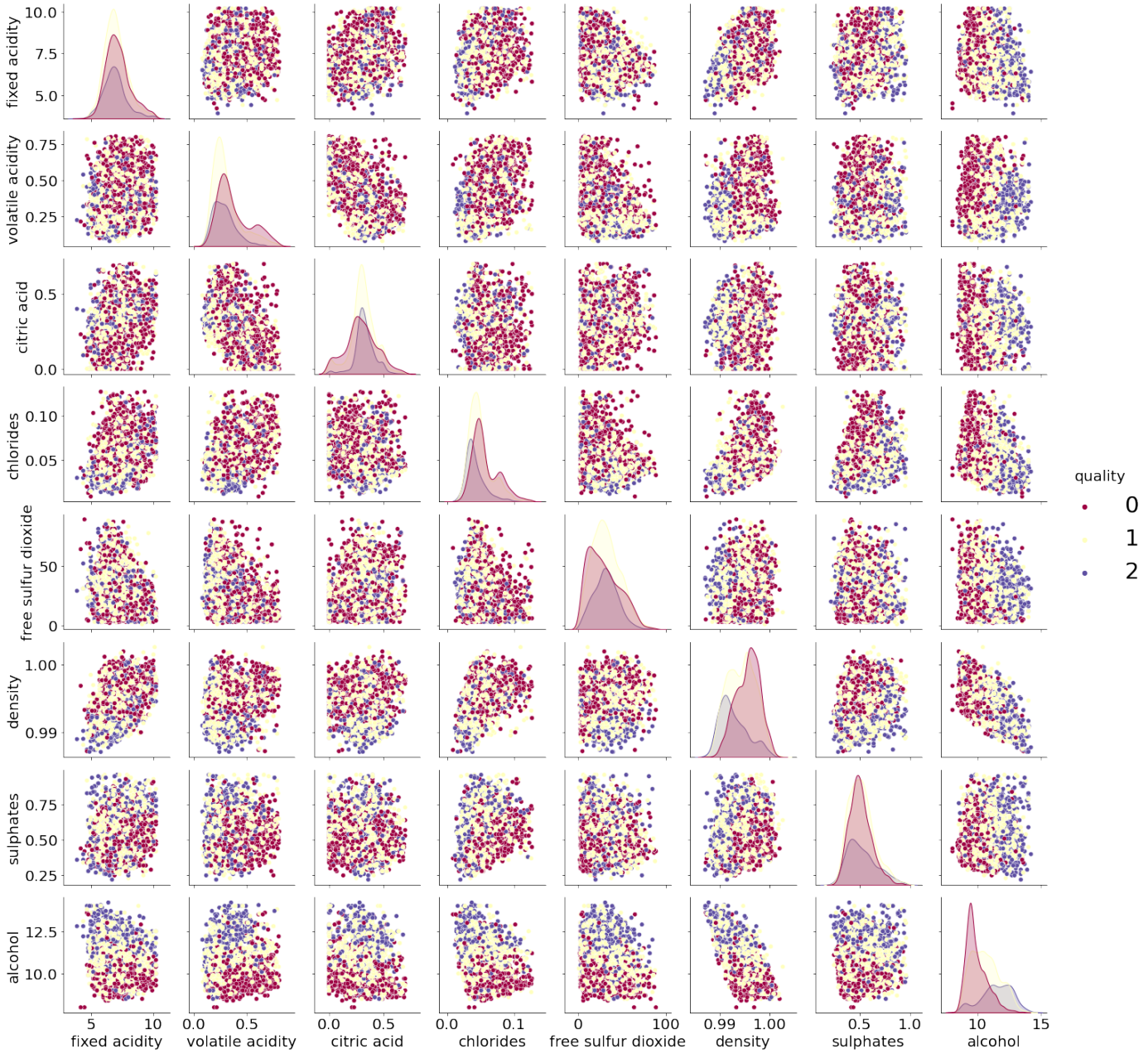
```
0 2084
1 2556
2 1192
```

При реализации и тестировании алгоритмов было замечено, что мульти-классовая классификация на три класса показала на некоторых алгоритмах маленькую точность, так как данные вина «среднего качества», находящиеся в первом и во втором классах могли не сильно отличаться друг от друга. Поэтому было принято решение насильно разделить данные на два класса, чтобы сравнить и проанализировать работу алгоритмов:

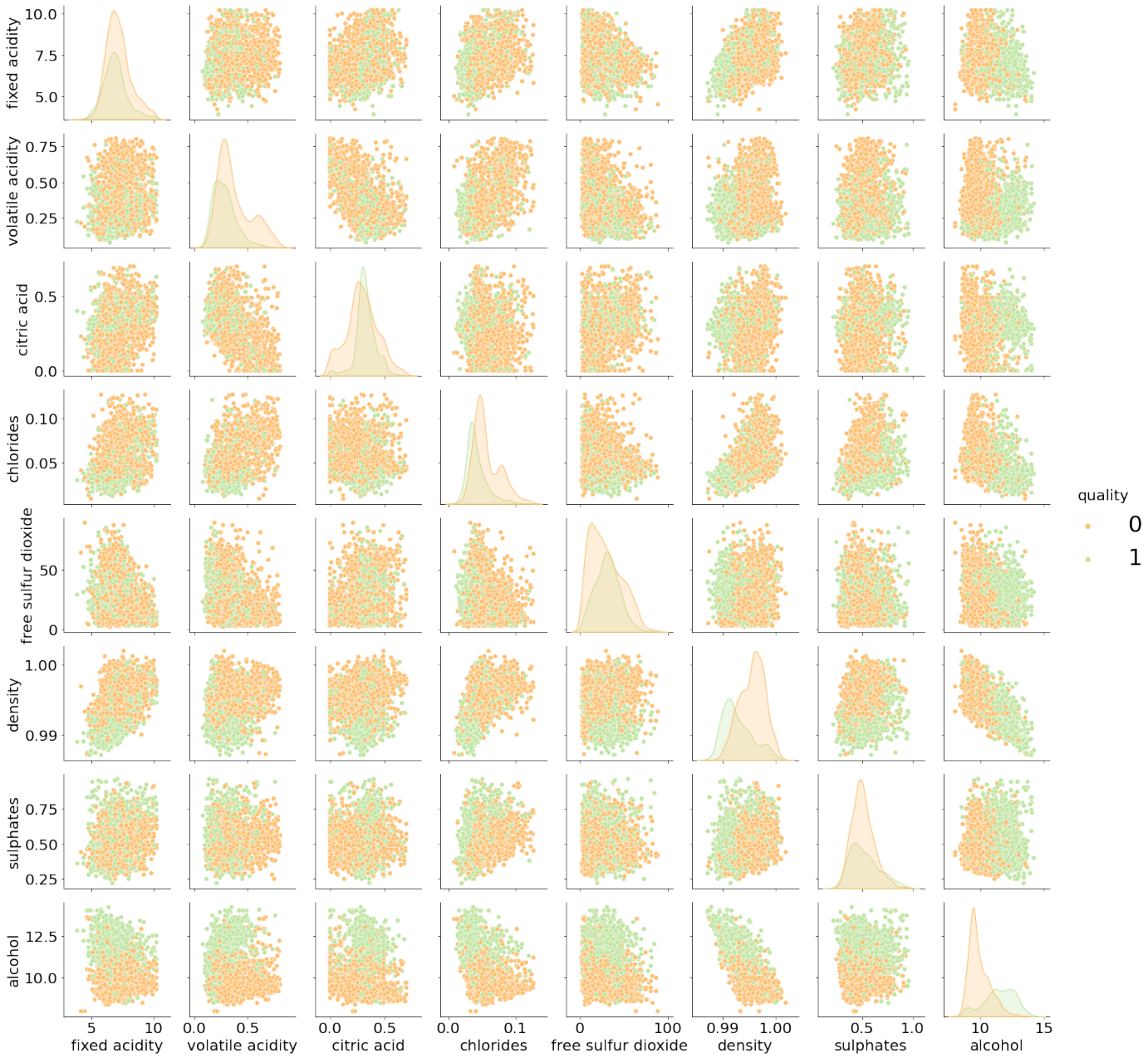
```
0 2084
1 1192
```

Посмотрим на зависимость данных:

Для трех классов:



Для двух классов:

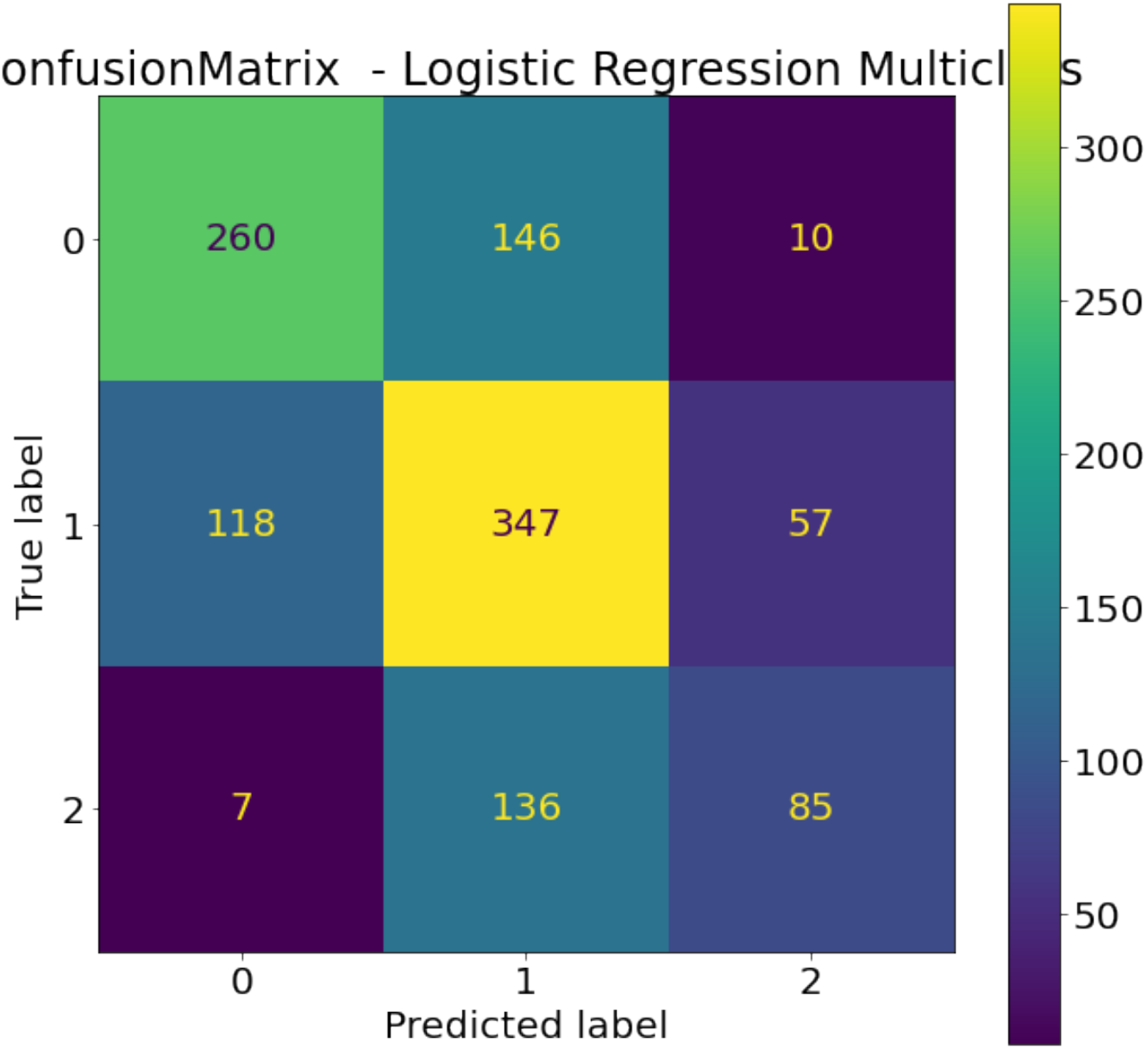


Как можно увидеть из графиков данные, насильно разделенные на два класса сильнее «разделяются» на графиках зависимости каждого параметра. Можно предположить, что алгоритмы для этих данных могут работать точнее.

Алгоритмы

Logistic Regression (Multiclass)

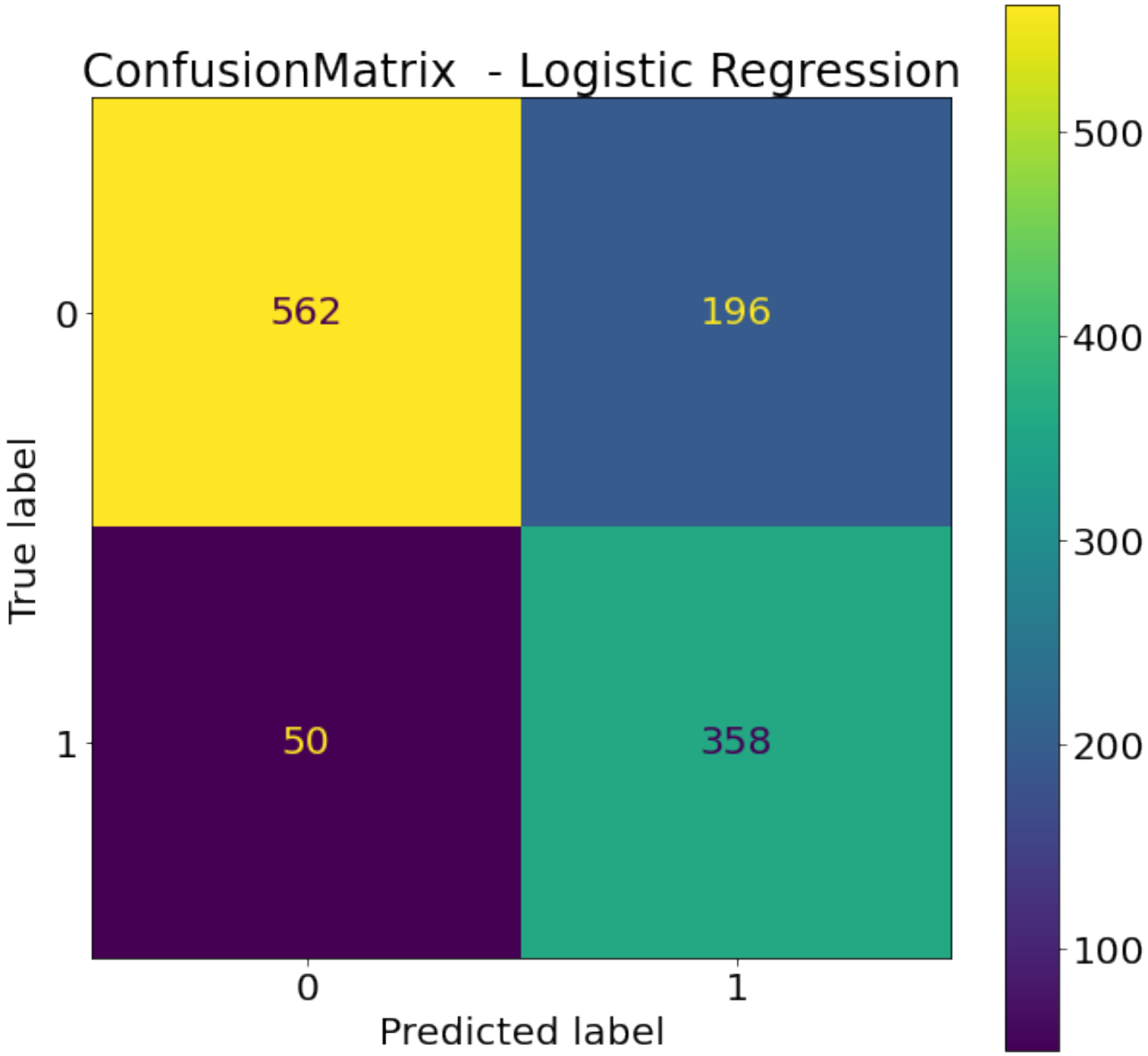
```
Best parameters:  {'LR__batch_size': 40, 'LR__countClasses': 3, 'LR__epoch': 80, 'LR__lr': 0.01}
Accuracy: 0.5934819897084048
Recall: 0.554185991799422
Precision: 0.5954015060052211
```



Анализируя результаты алгоритма, можно сделать вывод о том, что данные плохо линейно разделимы. Как можно увидеть на графиках зависимостей признаков, характеризующих данные, данные 0-го, 1-го и 2-го класса на большинстве графиков сильно перемешаны. Также по Confusion Matrix видно, что чаще всего ошибочные значения принадлежат 1-му класса, но и верных ответов больше всего у 1-го класса, это происходит потому, что при разделении данных на три более-менее сбалансированных класса, количество данных в 1-м классе все равно оказалось большим, чем в остальных. (почти в два раза больше, чем во 2-м классе)

Logistic Regression

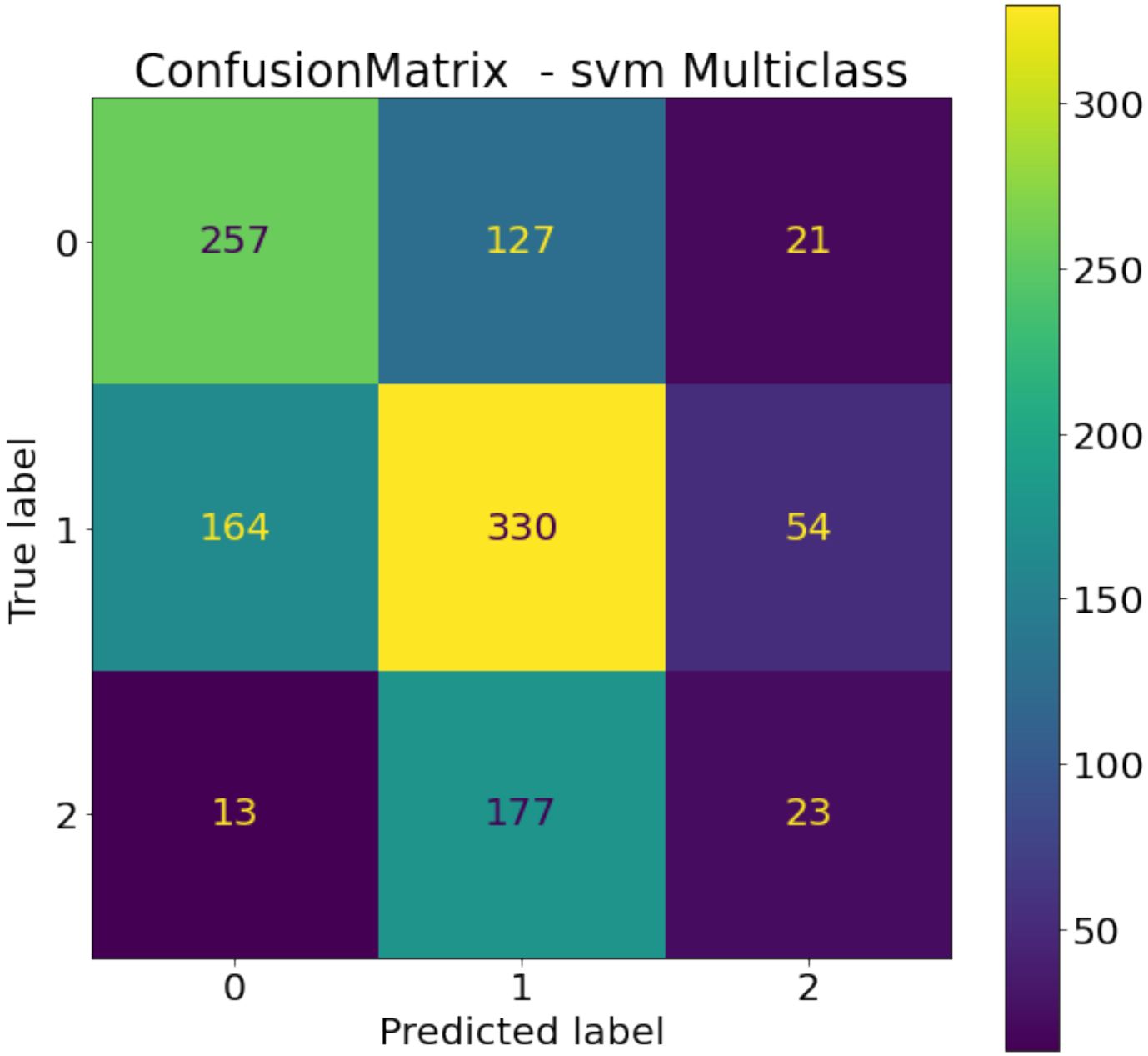
```
Best parameters:  {'LR2__batch_size': 50, 'LR2__epoch': 40, 'LR2__lr': 0.01}
Accuracy: 0.7890222984562607
Recall: 0.8094378912514875
Precision: 0.7822550199381798
```



Так как мульти-классификация не показала хорошую точность, проанализируем Логистическую Регрессию, классифицирующую два класса. При насильном разделении на два класса, чтобы данные были более-менее сбалансированными пришлось убрать один из классов, лежащих в середине, то есть выборка стала сильней линейно разделимой. Как можно увидеть, точность классификации возросла.

SVM (Multiclass)

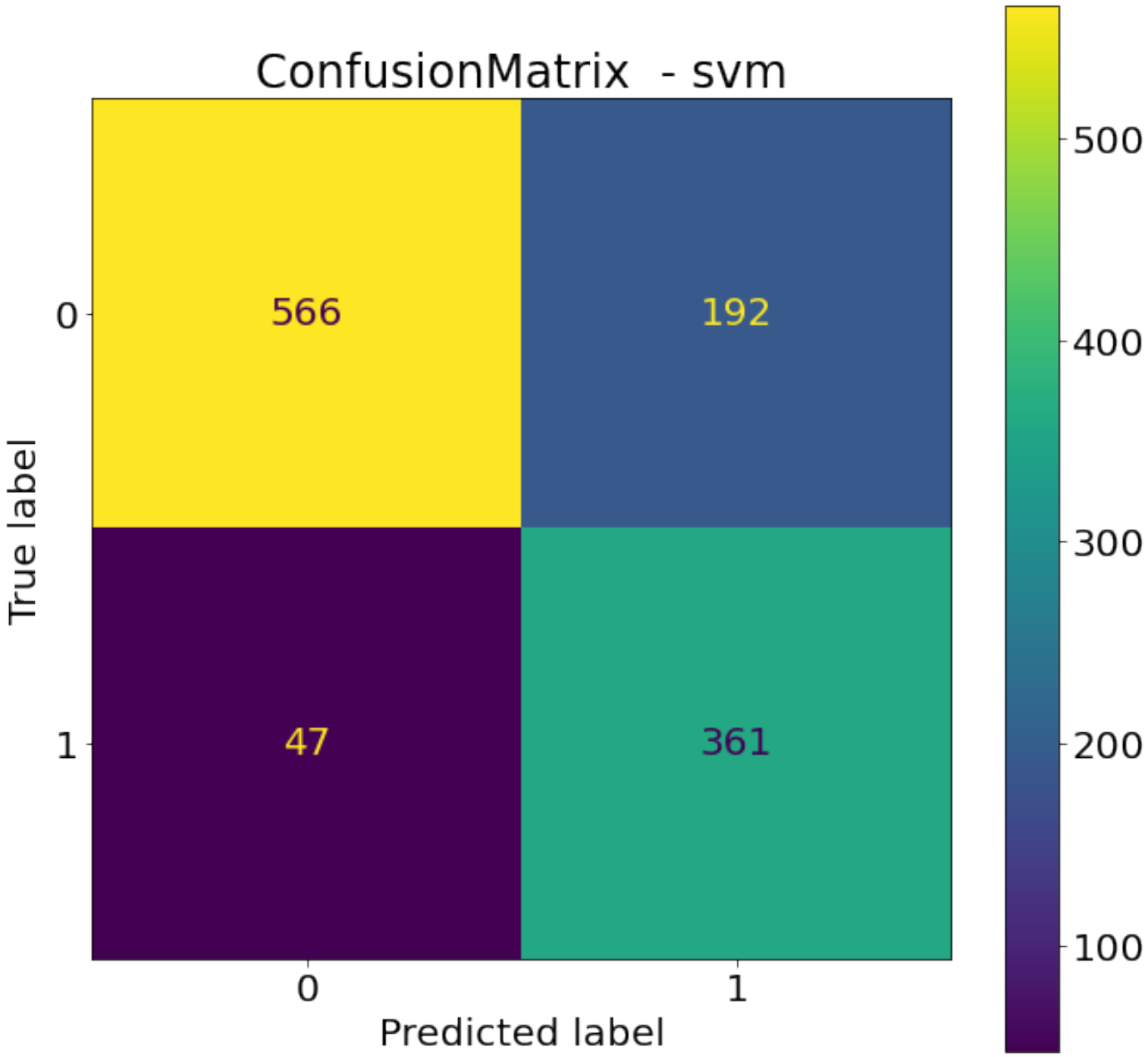
```
Best parameters:  {'SVM__alph': 0.1, 'SVM__batch_size': 10, 'SVM__epoch': 20, 'SVM__lr': 0.001}
Accuracy: 0.5231560891938251
Recall: 0.44824630097124757
Precision: 0.44912150267657686
```



Данный алгоритм, производящий классификацию трех классов, показал одну из самых маленьких точностей. Можно сделать вывод о том, что данные плохо разделяются n-мерной гиперплоскостью, сложно подобрать тако положение данных, чтобы при проекции они хорошо разделялись. Так как мульти-классовый алгоритм использует несколько (в данном случае 3) обычных SVM, которые пытаются отличить один класс от остальных, то точность алгоритма зависит от точности внутренних SVM. При анализе было выяснено, что точность SVM, которые отличают один класс от других, составляет от 55% до 75%, то можно сделать вывод о том, что такая маленькая точность вполне оправдана и происходит потому, что внутренние алгоритмы могут ошибаться, приводя мульти-классовый алгоритм к еще меньшей точности.

SVM

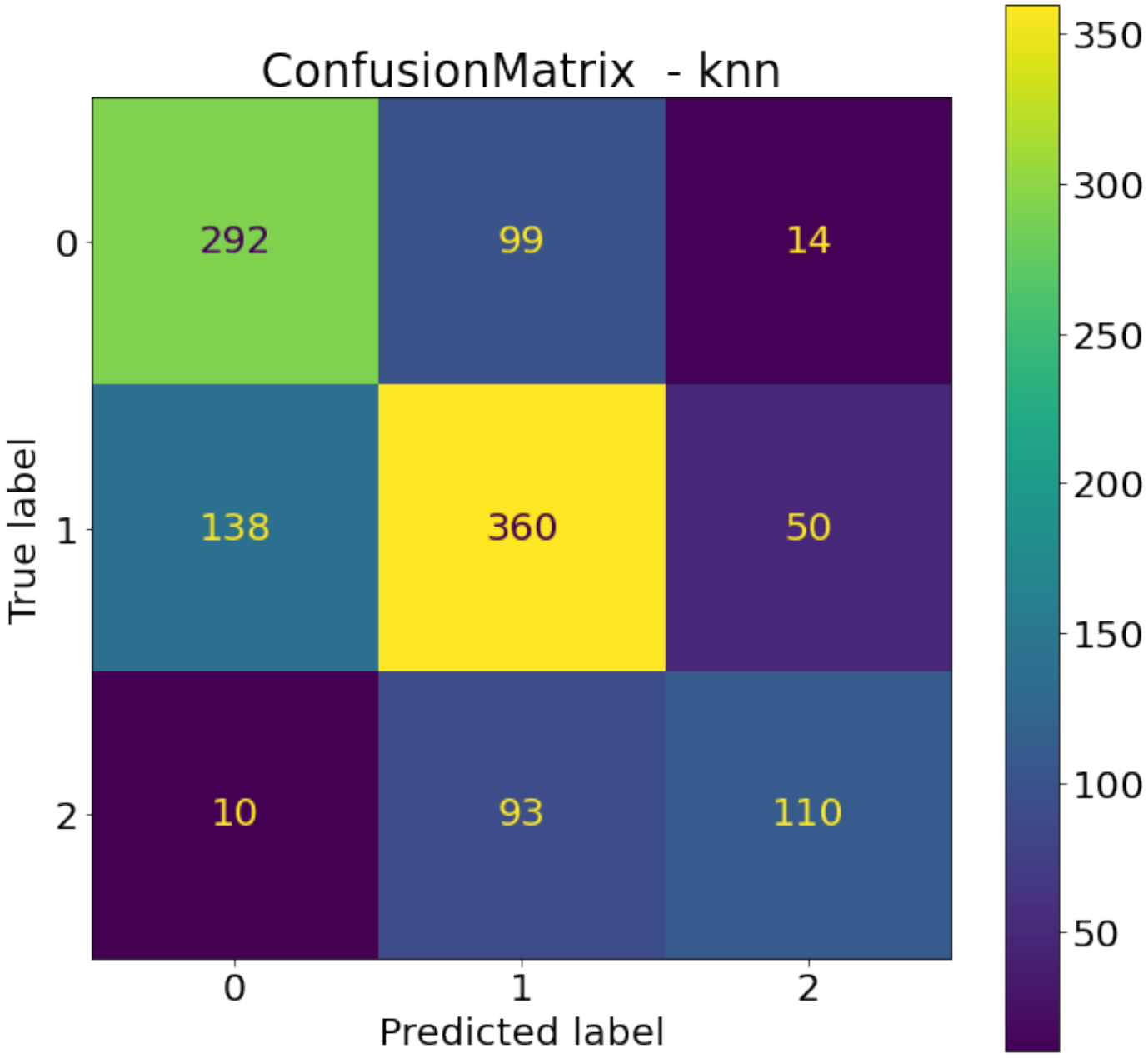
```
Best parameters:  {'SVM__alph': 0.1, 'SVM__batch_size': 5, 'SVM__epoch': 50, 'SVM__lr': 0.01}
Accuracy: 0.7950257289879932
Recall: 0.8157528842671633
Precision: 0.7880653944523274
```



Аналогично проанализируем разделение на два класса с помощью метода SVM. Точность стала намного выше, так как сами данные лучше разделимы. Можно также посмотреть на графики зависимостей характеристик для данных, на них наглядно можно увидеть «разделимость» данных и сравнить ее с аналогичными графиками для 3-х классов. Также можно сделать предположение об ошибочных утверждениях алгоритма, данные, разделенные на два класса, отличаются по размеру элементов. В 0-м классе почти в 2 раза больше элементов, что может быть одной из причин ошибок.

KNN

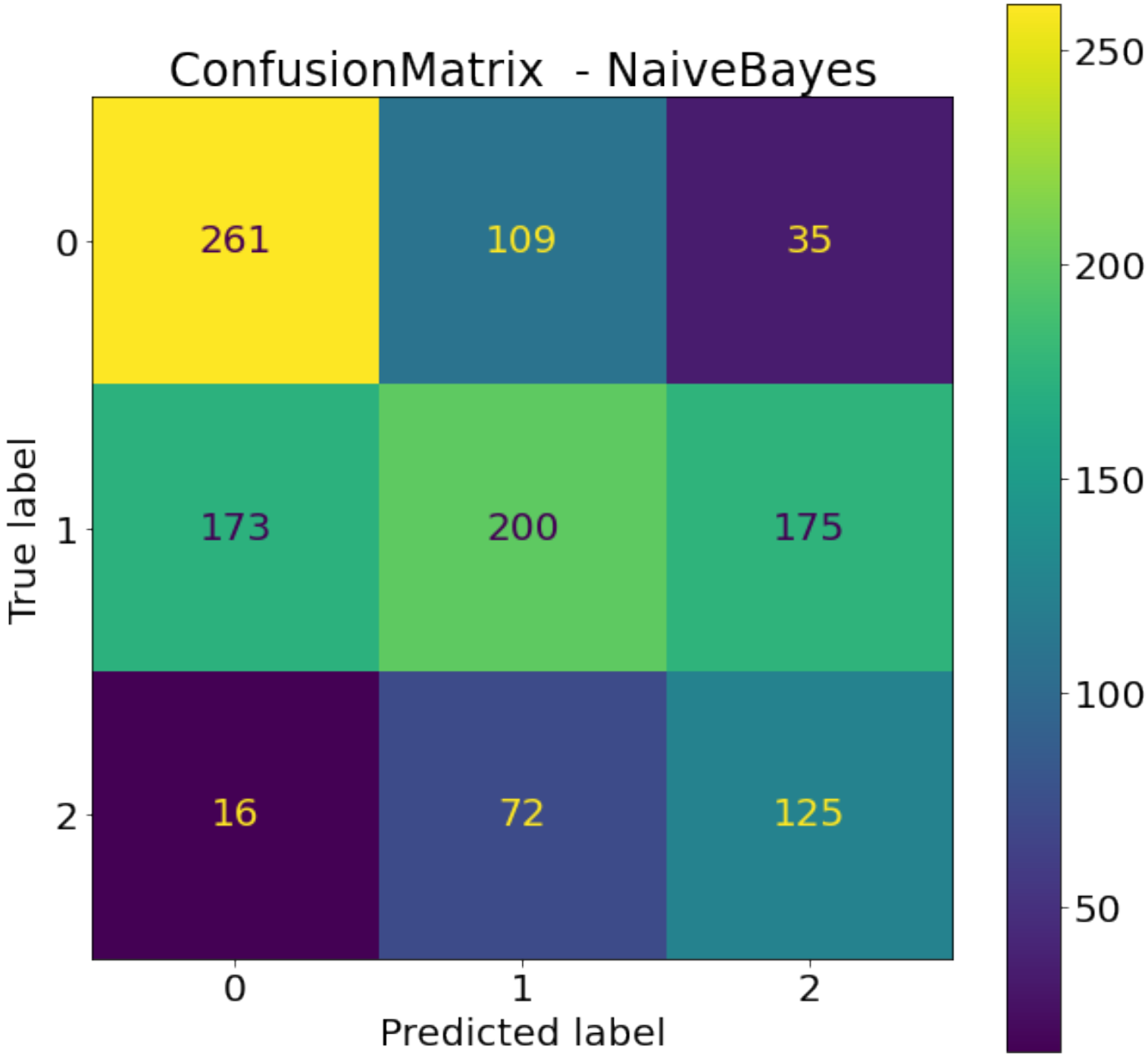
```
Best parameters:  {'KNN__countClasses': 3, 'KNN__k': 10}
Accuracy: 0.6535162950257289
Recall: 0.6314512952576532
Precision: 0.6493313949086064
```



Данный алгоритм, оказался наиболее точным при мульти-классовой классификации взятых мной данных. Это можно объяснить тем, что алгоритм не пытается «разделить» данные в прямом смысле, он делает свои предположения, основываясь на расстоянии до ближайших соседей. Аналогичные проблемы с точностью могут возникать из-за плохой балансировки данных.

Naive Bayes (for 3 classes)

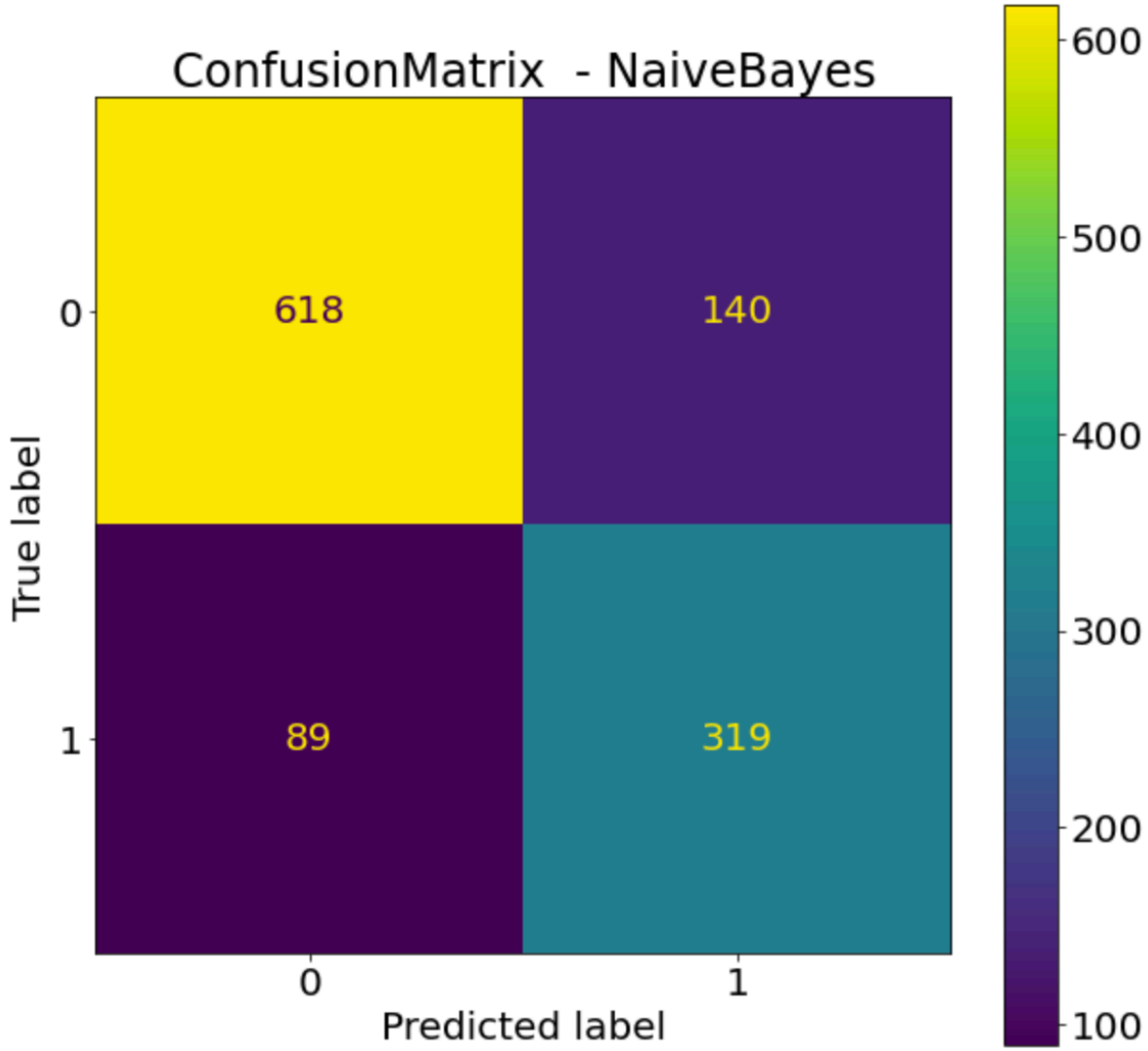
```
Best parameters:  {'Bayes__countClasses': 3, 'Bayes__lambd': 1}
Accuracy: 0.5025728987993139
Recall: 0.5320874693959922
Precision: 0.49268957052010287
```



По данному алгоритму можно оценить предположение о плохой точности при дисбалансе данных. Алгоритм Байеса делает свои предположения, основываясь на частоте встречаемости класса. По зеленой линии в центре Confusion Matrix видно, что алгоритм чаще всего при ошибках выдает результат 1-го класса. Это происходит потому, что изначально вероятность встретить 1-й класс намного выше, чем остальные. Вероятности встречаемости классов: 1-й класс – 0.35; 2-й класс – 0.44; 3-й класс – 0.21

Naive Bayes (for 2 classes)

```
Best parameters:  {'Bayes__countClasses': 2, 'Bayes__lambd': 1}
Accuracy: 0.8036020583190394
Recall: 0.7985830875885974
Precision: 0.7845525448903434
```

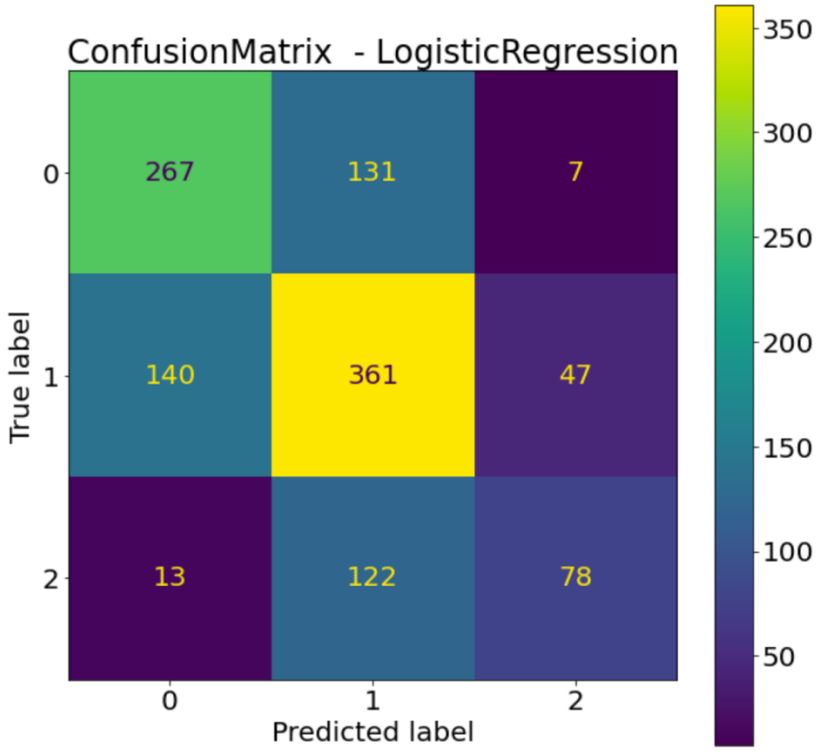


Как и в остальных случаях данный алгоритм для двух классов показал более высокую точность.

Sklearn

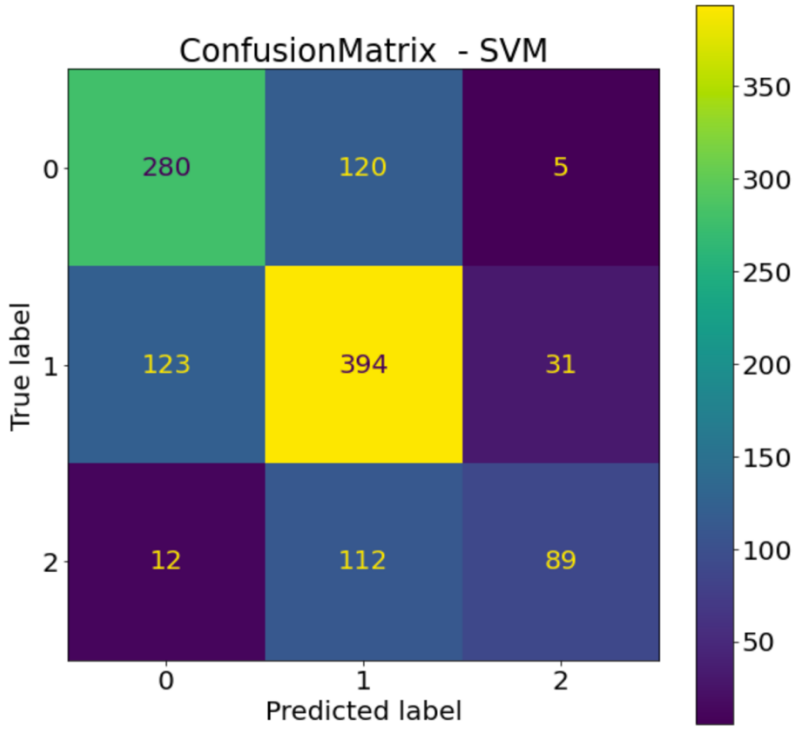
Logistic Regression

Accuracy: 0.6054888507718696
Recall: 0.5614051888151473
Precision: 0.6048570864531776



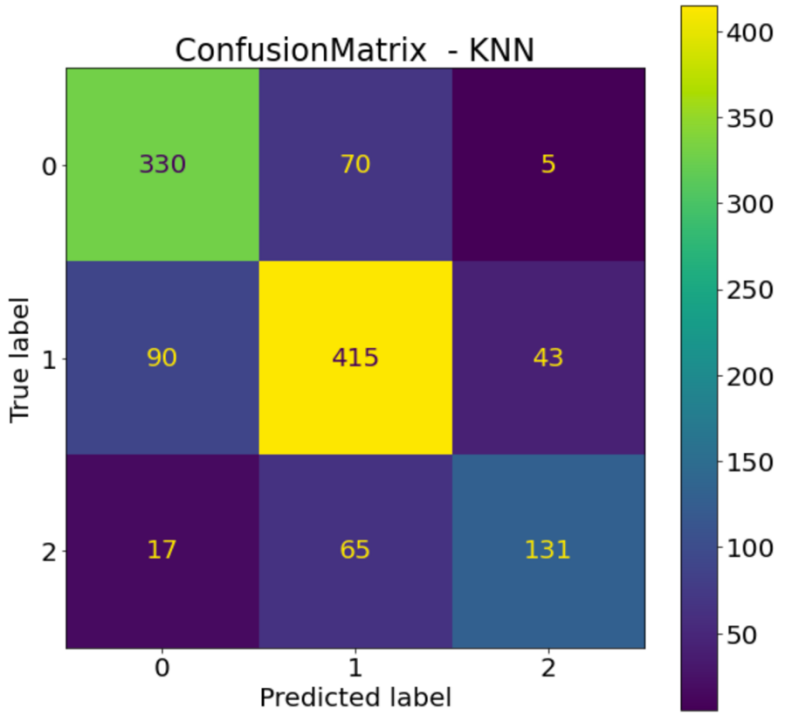
SVM

Accuracy: 0.6543739279588336
Recall: 0.6093921674893311
Precision: 0.6720305888089099



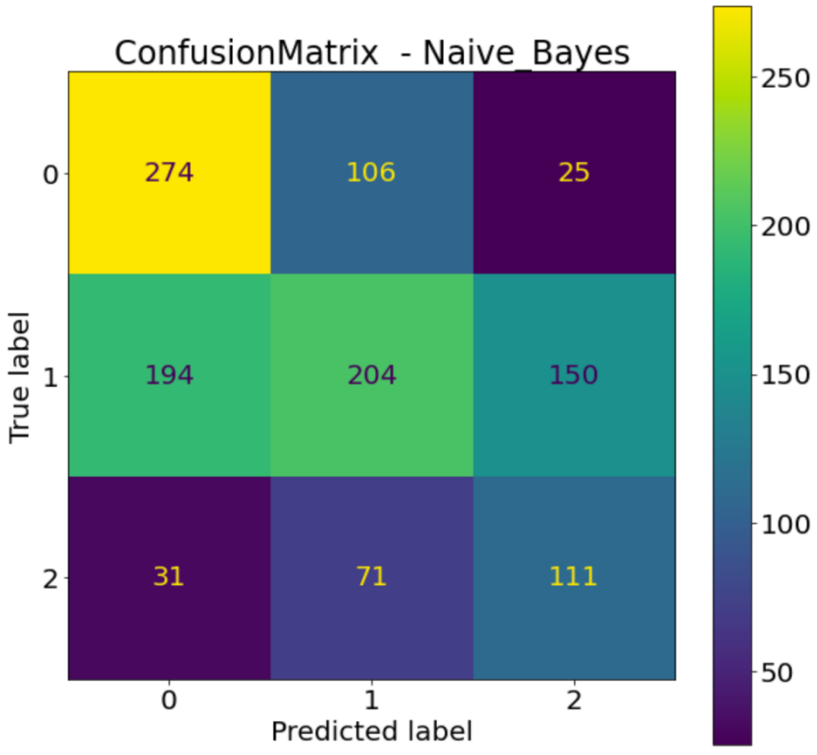
KNN

```
Accuracy: 0.7512864493996569
Recall: 0.7290458530220704
Precision: 0.7471792571277377
```



Naive Bayes

```
Accuracy: 0.5051457975986278
Recall: 0.5233109147208505
Precision: 0.4908810517902718
```



При сравнении точности алгоритмов из Sklearn с точностью реализованных алгоритмов можно увидеть, что точность первых в некоторых алгоритмах достаточно превышает точность реализованных мной алгоритмов, а в некоторых остается почти такой же. Так как точность данных алгоритмов не сильно высокая, можно подтвердить, представленные выше, некоторые предположения о данных.

Вывод:

В данной лабораторной работе я реализовала алгоритмы Logistic Regression, SVM, KNN, Naive Bayes для мульти-классовой классификации, а также для классификации двух классов. Получила точность данных алгоритмов, проанализировала ее, а также использовала реализацию данных алгоритмов из sklearn, также получив их точность. Данная лабораторная работа заставила меня задуматься о том, насколько важно качество данных, используемых для предсказания результата.

Ссылка на GitHub: https://github.com/Ler-B/AI