|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_РТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИУ-5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***\_\_\_\_\_\_\_\_\_Решение комплексной задачи \_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_машинного обучения\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_для дисбалансированной выборки\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_\_РТ5-61Б\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Забурунов Л. В.\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Гапанюк Ю. Е.\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2021*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине \_\_\_\_\_\_\_\_\_\_\_\_\_\_Технологии Машинного Обучения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы РТ5-61Б Забурунов Леонид Вячеславович

Тема курсовой работы: Решение комплексной задачи машинного обучения для дисбалансированной выборки

Направленность КР (учебная, исследовательская, практическая, производственная, др.) \_учебная\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_\_кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения работы: 25% к 3 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

***Задание:*** Для выбранного студентом набора данных провести типовое исследование в рамках задачи машинного обучения: анализ набора данных и внесение необходимых изменений в его структуру, выбор критериев и испытываемых моделей МО, подбор гиперпараметров, получение оптимальное модели и составлеине выводов.

***Оформление курсовой работы:***

Расчетно-пояснительная записка на \_\_\_ листах формата А4.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата выдачи задания** 16 февраля 2021 г.

**Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Гапанюк Ю. Е.\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Забурунов Л. В.\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Оглавление**

[Введение 4](#_Toc74092117)

[1. Разведочный анализ данных 5](#_Toc74092118)

[2. Восстановление баланса выборки 12](#_Toc74092119)

[3. Подготовка данных и конвейеров для обучения 14](#_Toc74092120)

[4. Обучение моделей 17](#_Toc74092121)

[4.1 Логистическая регрессия 17](#_Toc74092122)

[4.2 Машина опорных векторов 24](#_Toc74092123)

[4.2.1 Линейная модель 24](#_Toc74092124)

[4.2.2 Полиномиальная модель 30](#_Toc74092125)

[4.3 Дерево решений 34](#_Toc74092126)

[4.4 Случайный лес 37](#_Toc74092127)

[4.5 Градиентный бустинг 39](#_Toc74092128)

[Заключение 41](#_Toc74092129)

[Список литературы 42](#_Toc74092130)

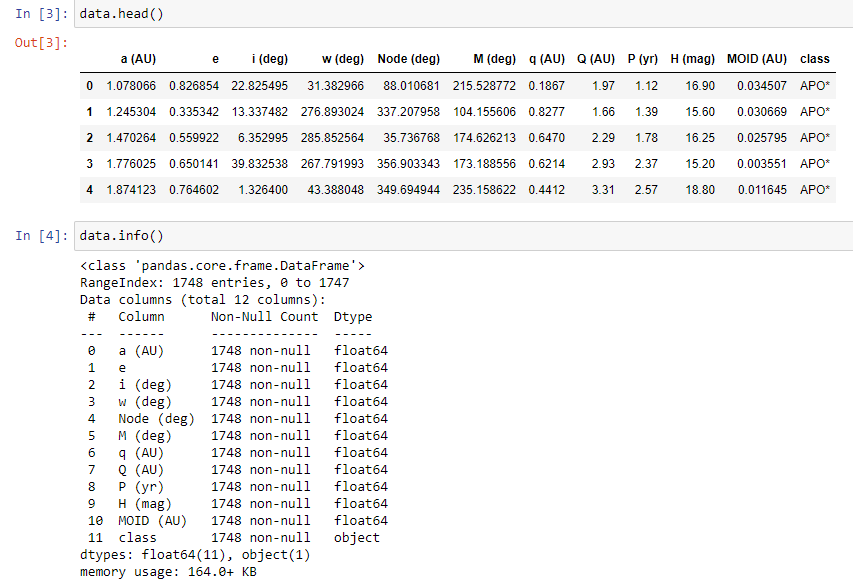
# Введение

Проблема дисбаланса классов занимает отдельное место в науке о данных; если брать шире, то проблема нехватки данных занимает отдельное место в математической статистике. Для восстановления баланса предлагается манипулировать над набором данных: удалять имеющиеся или искусственно создавать новые.

В рамках данного курсового проекта проводится полное исследование набора данных и возможностей для предсказания новых после обучения. При этом отдельное, опять же, место отводится изучению влияния методов восстановления баланса выборки. Для исследования взят набор данных для многоклассовой классификации по предсказанию типов орбит небесных тел на основе измеренных величин: <https://www.kaggle.com/brsdincer/orbitclassification>. Для набора данных были выбраны признаки для обучения, проведено масштабирование данных, их обработка с точки зрения восстановления баланса и обучение различных моделей. Возможности для исследования данной выборки иссякли раньше ожидаемого, поэтому часть работы проводилась на использовавшемся в лабораторных работах по данному курсу наборе данных по предсказанию пульсаров среди нейтронных звёзд: <https://www.kaggle.com/colearninglounge/predicting-pulsar-starintermediate>.

# Разведочный анализ данных

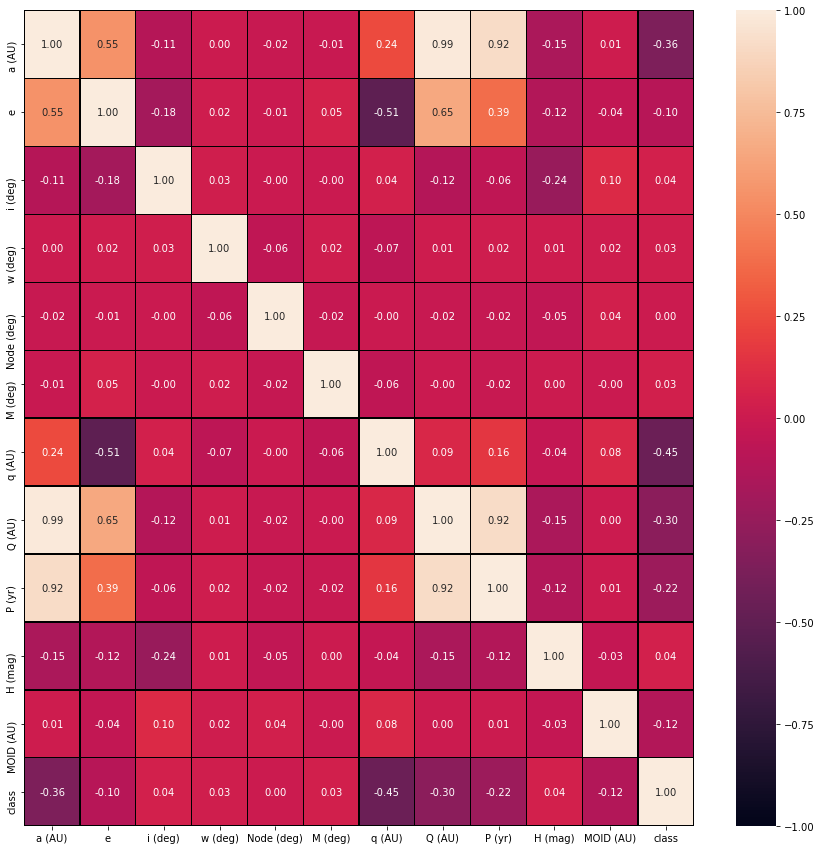
Изначально выборка предоставлена нам в следующем виде:



Набор данных имеет следующие колонки:

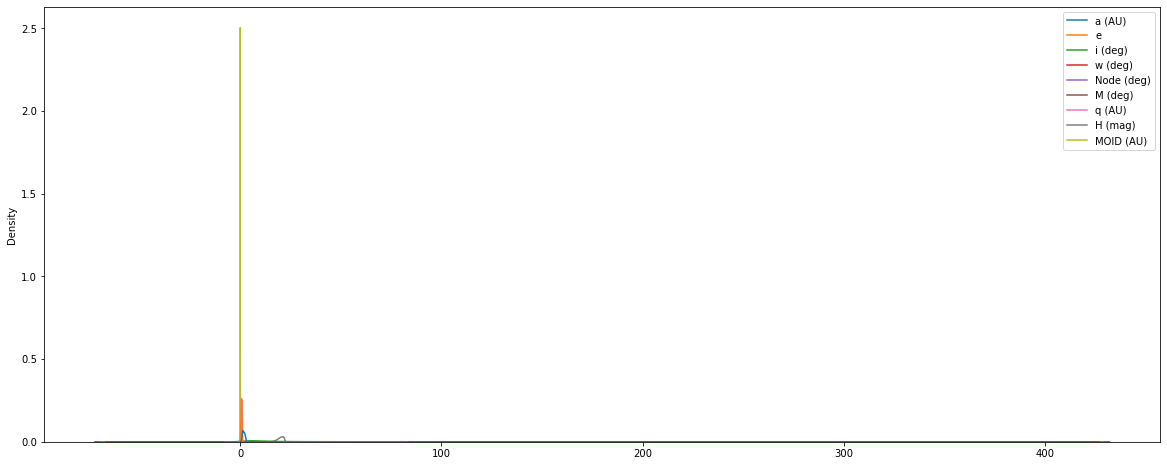
1. *a (AU)* - длина большей полуоси орбиты, AU - астрономические единицы;
2. *e* - эксцентрисцитет орбиты, безразмерная величина;
3. *i (deg)* - наклон орбиты в эклиптической системе координат, deg - градусы;
4. *w (deg)* - перигелий;
5. *Node (deg)* - долгота восходящего узла;
6. *M (deg)* - средняя аномалия в эпоху;
7. *q (AU)* - длина орбиты в перигелий;
8. *Q (AU)* - длина орбиты в афелий;
9. *P (yr)* - период обращения, *yr* - год по Юлианскому (человеческому) календарю;
10. *H (mag)* - абсолютная V-длина, mag - сокращание "magnitude";
11. *MOID (AU)* – «минимальное расс»;
12. *class* - строковая метка класса небесного тела.

Корреляционная матрица:

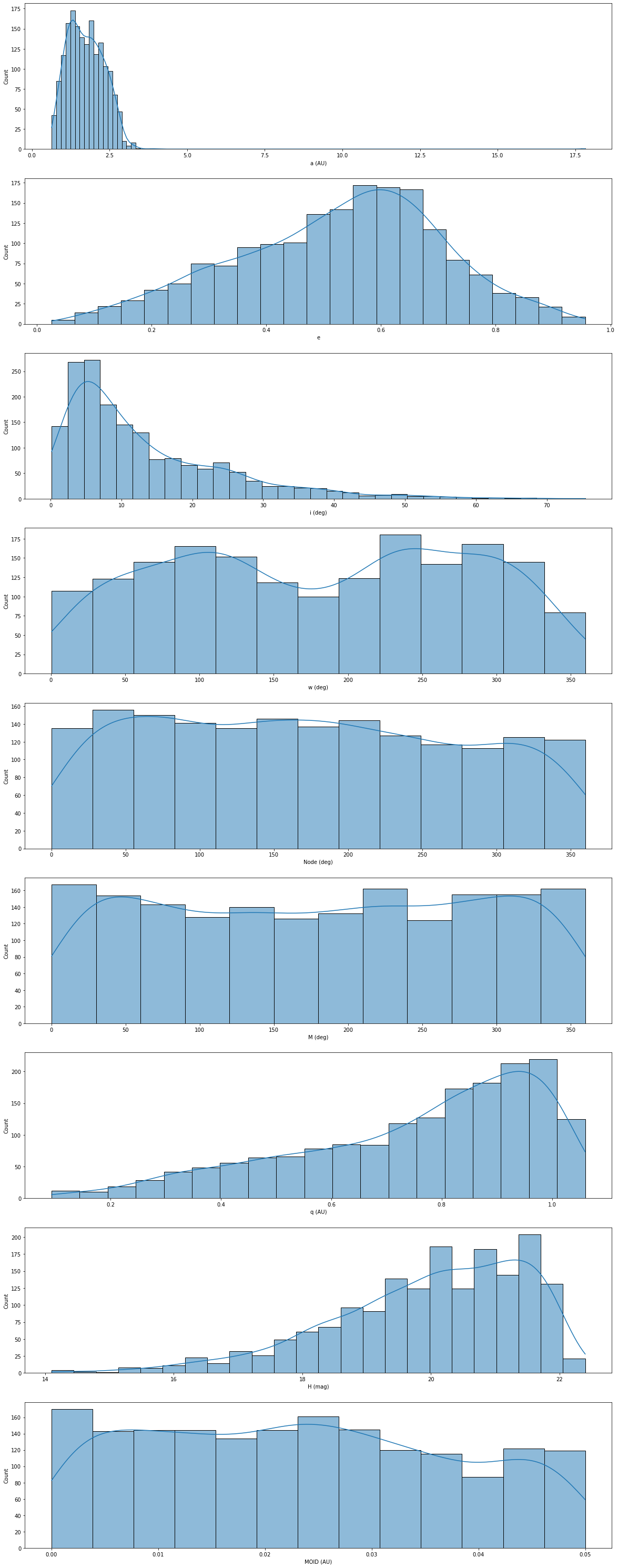


Имеется треугольник из сверхсильных линейных зависимостей между **a**, **Q** и **P**. Чтобы избавиться от нежелательного влияния таких строгих линейных зависимостей на качество модели, удалим признаки **Q** и **P**, поскольку они зависят от целевого меньше, нежели **a**.

Распределение оставшихся признаков на одном графике:



И для каждой колонки по отдельности:



Также видим одиночную аномалию для первого признака:

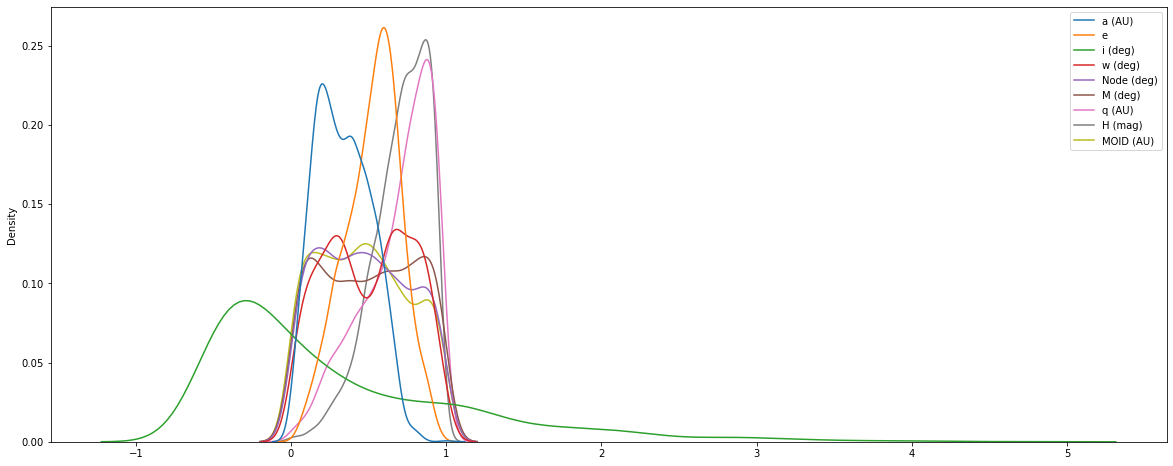
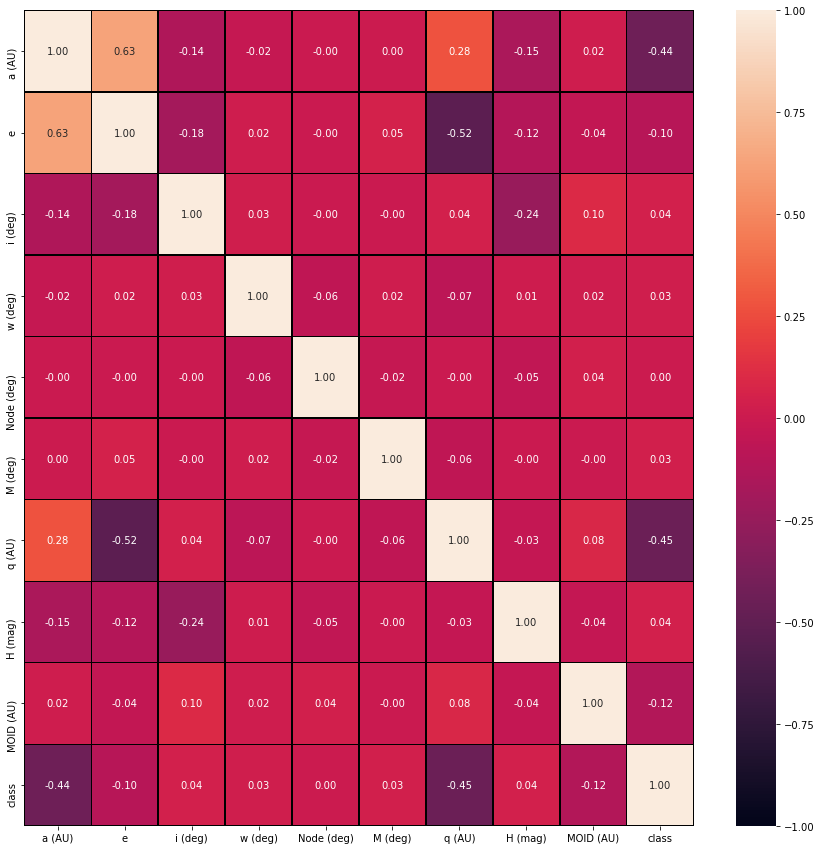


Заметим, что данная аномалия принадлежит к доминирующему классу, поэтому можно удалить данную запись.

Итак, с точки зрения масштабирования видим следующее:

1. Признак **e** уже распределён в интервале [0; 1] и не требует масштабирования, что не является сюрпризом, поскольку эксцентриситет эллиптического сечения не может принимать другие значения;
2. Признак **i** имеет нормальное распределение с длинным хвостом. Применим для него *RobustScaler*(), чтобы слишком большое количество примеров не "смялось" в близких к нулю значениях и в то же время интервал распределения как можно меньше отличался от остальных признаков.  
   Можно заметить, что **i** измеряется в градусах, и с точки зрения человеческого понимания правильное решение - это, возможно, масштабировать вместе с остальными признаками, имеющими значения в градусах. Однако модели машинного обучения человеческое понимание не слишком интересно, поэтому не будем обращать внимание на единицы измерения;
3. Остальные признаки, выраженные в градусах - **w**, **Node** и **M**, - заключены в интервале [0; 360], поэтому для них применим мин-макс масштабирование;
4. Признаки **a**, **q**, **H** и **MOID** тоже заключены в строгом интервале, поэтому тоже используем мин-макс масштабирование.

По завершении масштабирования и, соответственно, всей предобработки получаем следующую структуру выборки:



# Восстановление баланса выборки

Наша выборка – крайне несбалансированная. Распределение числа записей по классам:

1. Класс 0 – 96 (5.5%);
2. Класс 1 – 14 (0.8%);
3. Класс 2 – 1476 (84.5%);
4. Класс 3 – 7 (0.4%);
5. Класс 4 – 149 (8.5%);
6. Класс 5 – 5 (0.2%).

Для восстановления баланса выборки существует две основных группы методов: *undersampling* (исключение записей, относящихся к доминирующему классу) и *oversampling* (искусственное внесение данных, относящихся к тем классам, для которых не хватает данных). Методы обеих групп могут оказаться полезными, однако больший прирост качества ожидается для второй группы потому, что 5, 7 и 14 образцов - это катастрофически малое значение.

Воспользуемся следующими методами восстановления баланса:

1. *Random Undersampling* (удаление случайных элементов до тех пор, пока не будет достигнуто желаемое соотношение);
2. *Edited Nearest Neighbours* (снижение числа объектов доминирующего класса на разделяющей полосе засчёт работы KNN);
3. *Instance Hardness Threshold* (исключение объектов, предсказание в отношении которых - самое неуверенное);
4. SMOTE (*Synthetic Minority Over-Sampling Technique*; интерполяция ближайших соседей для малопредставленных классов);
5. ADASYN (*Adaptive Synthetic Algorithm*, расширение SMOTE).

Общий процесс обучения будет следующим:

1. Сформируем несколько дополнительных *DataFrame*-ов, в которых будут содержаться более сбалансированные данные;
2. При обучении каждой модели подбор гиперпараметров будет происходить на основном наборе;
3. Для сравнения будем смотреть на результаты лучшей модели со всеми наборами данных.

# 3. Подготовка данных и конвейеров для обучения

Разбиваем все *DataFrame*-ы на обучающую и проверочную выборки:

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(data[columns[:**-**1]],

data[columns[**-**1]], train\_size **=** 0.75, random\_state **=** 12)

rus\_x\_train, rus\_x\_test, rus\_y\_train, rus\_y\_test **=** train\_test\_split(data\_rus[columns[:**-**1]], data\_rus[columns[**-**1]], train\_size **=** 0.75, random\_state **=** 12)

enn\_x\_train, enn\_x\_test, enn\_y\_train, enn\_y\_test **=** train\_test\_split(data\_enn[columns[:**-**1]], data\_enn[columns[**-**1]], train\_size **=** 0.75, random\_state **=** 12)

iht\_x\_train, iht\_x\_test, iht\_y\_train, iht\_y\_test **=** train\_test\_split(data\_iht[columns[:**-**1]], data\_iht[columns[**-**1]], train\_size **=** 0.75, random\_state **=** 12)

smote\_x\_train, smote\_x\_test, smote\_y\_train, smote\_y\_test **=** train\_test\_split(data\_smote[columns[:**-**1]], data\_smote[columns[**-**1]], train\_size **=** 0.75, random\_state **=** 12)

Подготавливаем методы для обучения и вывода показателей качества моделей:

* Прячем сюда "рутинную" работу по обучению моделей

**def** FitPredictClassifier(model, x\_train, x\_test, y\_train, y\_test):

solver **=** model

solver**.**fit(x\_train, y\_train)

prediction **=** solver**.**predict(x\_test)

**return** solver, prediction

* Прячем сюда "рутинную" работу по выводу результатов обучения

**def** PrintClassificationMetrics(y\_test, y\_predicted):

* + Результат accuracy не требует никакой обработки accuracy **=** accuracy\_score(y\_test, y\_predicted) print("1. Общая точность (accuracy) =", accuracy)
  + А вот результаты по precision и recall требуют обработки, поскольку функция вернёт список значений по к scores\_list **=** precision\_score(y\_test, y\_predicted, zero\_division **=** 0, average **= None**)

print("2. Меткость (precision) по классам:", FormatMulticlassMetricsScore(scores\_list, "precision"))

scores\_list **=** recall\_score(y\_test, y\_predicted, average **= None**)

print("3. Полнота (recall) по классам:", FormatMulticlassMetricsScore(scores\_list, "recall"))

**return**

* Упаковка оценки качества обучения на многоклассовой классификации в строковый формат для чтения

**def** FormatMulticlassMetricsScore(scores\_list, metric\_label):

result\_string **=** str() i **=** 0

sum **=** 0

**for** score **in** scores\_list:

result\_string **+=** "\nКласс {0}: {1}\_score = {2}"**.**format(i, metric\_label, score)

i **=** i **+** 1

sum **=** sum **+** score

result\_string **+=** "\nСреднее арифм-е: {0}\_score = {1}"**.**format(metric\_label, sum **/** i)

weights **=** np**.**array([data[columns[**-**1]]**.**value\_counts(sort **= False**)], dtype **=** float)

**for** i **in** range(weights**.**shape[1]):

weights[0, i] **/=** data[columns[**-**1]]**.**shape[0]

result\_string **+=** "\nСредневзвешенное: {0}\_score = {1}"**.**format(metric\_label, (np**.**dot(weights, scores\_list) **return** result\_string

* Небольшая обёртка над выводом матрицы ошибок, чтобы не создавать каждый раз новую фигуру явно **def** ShowConfusionMatrix(fitted\_model, x\_test, y\_test):

fig, ax **=** plt**.**subplots(figsize **=** (8, 8))

plot\_confusion\_matrix(fitted\_model, x\_test, y\_test, ax **=** ax)

**return**

**def** PlotMulticlassRocAuc(base\_model, x\_train, x\_test, y\_train, y\_test):

* Предварительно превращаем метки классов в своего рода One-Hot-Encoded таблицу y\_train **=** label\_binarize(y\_train, classes **=** sorted(y\_train**.**unique()))

y\_test **=** label\_binarize(y\_test, classes **=** sorted(y\_test**.**unique()))

* Обучаем классификатор

ovrc **=** OneVsRestClassifier(base\_model)

y\_score **=** ovrc**.**fit(x\_train, y\_train)**.**decision\_function(x\_test)

n\_classes **=** y\_score**.**shape[1]

* Вычисление ROC и AUC-ROC fpr **=** dict()

tpr **=** dict() roc\_auc **=** dict()

* Строим кривую и считаем площадь под ней для каждого класса **for** i **in** range(n\_classes):

fpr[i], tpr[i], \_ **=** roc\_curve(y\_test[:, i], y\_score[:, i]) roc\_auc[i] **=** auc(fpr[i], tpr[i])

* Настраиваем область отрисовки

fig, ax **=** plt**.**subplots(figsize **=** (10, 10))

ax**.**set\_title("ROC AUC ({})"**.**format(base\_model))

ax**.**set\_xlim(0.0, 1.0)

ax**.**set\_ylim(0.0, 1.01)

ax**.**set\_xlabel("FPR")

ax**.**set\_ylabel("TPR")

* Рисуем базовую кривую, для которой AUC-ROC = 0.5 ax**.**plot([0, 1], [0, 1], 'k--')
* Отрисовываем кривую для каждого класса

**for** i **in** range(n\_classes):

ax**.**plot(fpr[i], tpr[i], label**=**'ROC curve for class {0} (area = {1})'**.**format(i, roc\_auc[i]))

ax**.**legend(loc**=**"lower right")

**return**

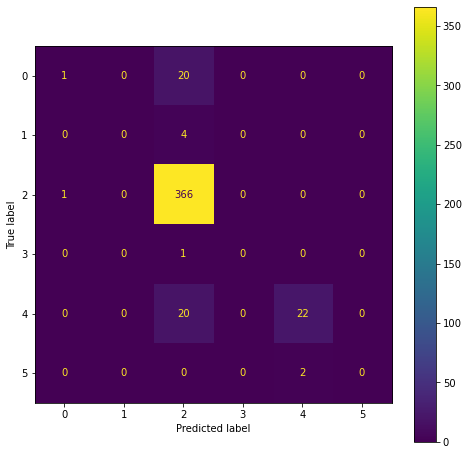
# Обучение моделей

Воспользуемся следующими моделями:

1. Логистическая регрессия (используется стратегия *One-versus-Rest*);
2. *LinearSVC* (*OvR*);
3. *SVC* (*OvO*);
4. Дерево решений;
5. Случайный лес;
6. Градиентный бустинг.

# Логистическая регрессия

Обучая модель без какой-либо настройки параметров, получаем следующий результат:



Подбираем гиперпараметры модели с использованием *Randomized Search* и генерации распределений для получения оптимальной модели:

logreg\_grid = RandomizedSearchCV(estimator = LogisticRegression(penalty = "elasticnet", solver = "saga", max\_iter = 1e+04),

param\_distributions = {'C': expon(scale=100), 'l1\_ratio' : uniform()},

n\_iter = 20,

scoring = "precision\_macro",

n\_jobs = 8,

refit = True,

random\_state = 16,

error\_score = 0)

Получаем результат:

1. Общая точность (accuracy) = 0.9794050343249427

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 0.0

Класс 2: precision\_score = 0.9812834224598931

Класс 3: precision\_score = 1.0

Класс 4: precision\_score = 0.9534883720930233

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.6557952990921527

Средневзвешенное: precision\_score = 0.9693440749814899

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.9047619047619048

Класс 1: recall\_score = 0.0

Класс 2: recall\_score = 1.0

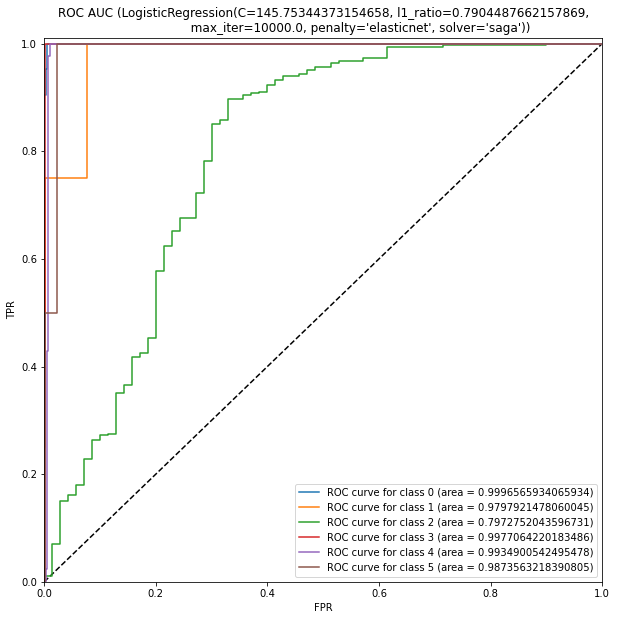
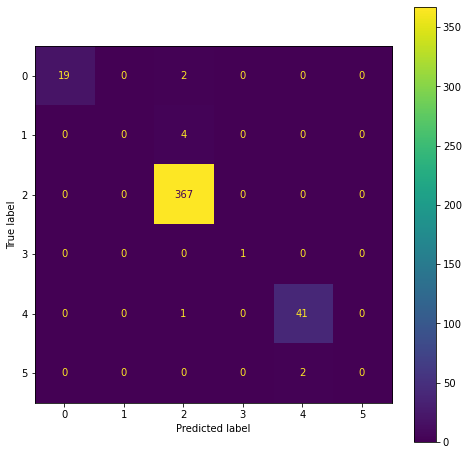
Класс 3: recall\_score = 1.0

Класс 4: recall\_score = 0.9761904761904762

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.6468253968253969

Средневзвешенное: recall\_score = 0.9818600594215935



Теперь по очереди используем методы восстановления баланса выборки:

1. RUS (*Random UnderSampling*):

1. Общая точность (accuracy) = 0.9378238341968912

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.9629629629629629

Класс 1: precision\_score = 1.0

Класс 2: precision\_score = 0.9523809523809523

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.8717948717948718

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.6311897978564646

Средневзвешенное: precision\_score = 0.9399291162313487

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.9285714285714286

Класс 1: recall\_score = 0.2

Класс 2: recall\_score = 0.9836065573770492

Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 1.0

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.5186963309914129

Средневзвешенное: recall\_score = 0.9689445539962117

1. ENN (*Edited Nearest Neighbours*):

1. Общая точность (accuracy) = 0.961218836565097

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 1.0

Класс 2: precision\_score = 0.9735973597359736

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.8604651162790697

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.6390104126691739

Средневзвешенное: precision\_score = 0.9589233001121228

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.7777777777777778

Класс 1: recall\_score = 0.25

Класс 2: recall\_score = 0.9932659932659933

Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 0.9736842105263158

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.4991213302616811

Средневзвешенное: recall\_score = 0.9669755123615879

1. IHT (*Instance Hardness Threshold*):

1. Общая точность (accuracy) = 0.9444444444444444

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.8

Класс 1: precision\_score = 0.0

Класс 2: precision\_score = 0.9790794979079498

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.875

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.4423465829846583

Средневзвешенное: precision\_score = 0.9457906919932079

3. Полнота (recall) по классам:

Класс 0: recall\_score = 1.0

Класс 1: recall\_score = 0.0

Класс 2: recall\_score = 0.9831932773109243

Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 0.9459459459459459

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.48818987054281177

Средневзвешенное: recall\_score = 0.9663075118814369

1. SMOTE (*Synthetic Minority OverSampling Technique*):

1. Общая точность (accuracy) = 0.9802130898021308

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.927536231884058

Класс 1: precision\_score = 0.9642857142857143

Класс 2: precision\_score = 0.9912023460410557

Класс 3: precision\_score = 0.9830508474576272

Класс 4: precision\_score = 1.0

Класс 5: precision\_score = 0.9672131147540983

Среднее арифм-е: precision\_score = 0.9722147090704256

Средневзвешенное: precision\_score = 0.9881371279470188

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.9552238805970149

Класс 1: recall\_score = 1.0

Класс 2: recall\_score = 0.9797101449275363

Класс 3: recall\_score = 1.0

Класс 4: recall\_score = 0.9594594594594594

Класс 5: recall\_score = 1.0

Среднее арифм-е: recall\_score = 0.9823989141640018

Средневзвешенное: recall\_score = 0.976939396628401

1. ADASYN (*Adaptive Synthenic Algorithm*):

1. Общая точность (accuracy) = 0.9774436090225563

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.953125

Класс 1: precision\_score = 0.9482758620689655

Класс 2: precision\_score = 0.9913544668587896

Класс 3: precision\_score = 0.9649122807017544

Класс 4: precision\_score = 0.9647058823529412

Класс 5: precision\_score = 0.9814814814814815

Среднее арифм-е: precision\_score = 0.9673091622439887

Средневзвешенное: precision\_score = 0.9865014453322535

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.953125

Класс 1: recall\_score = 1.0

Класс 2: recall\_score = 0.9745042492917847

Класс 3: recall\_score = 1.0

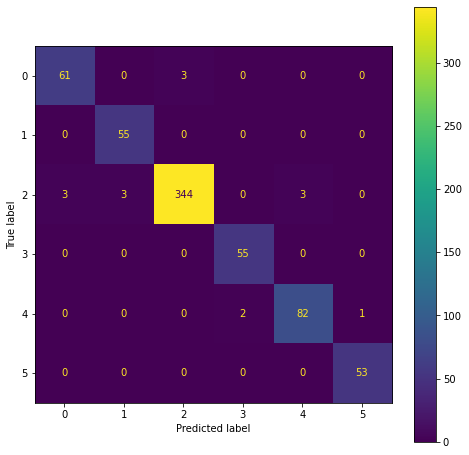
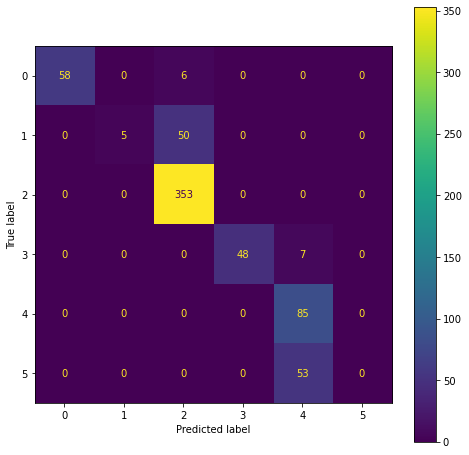
Класс 4: recall\_score = 0.9647058823529412

Класс 5: recall\_score = 1.0

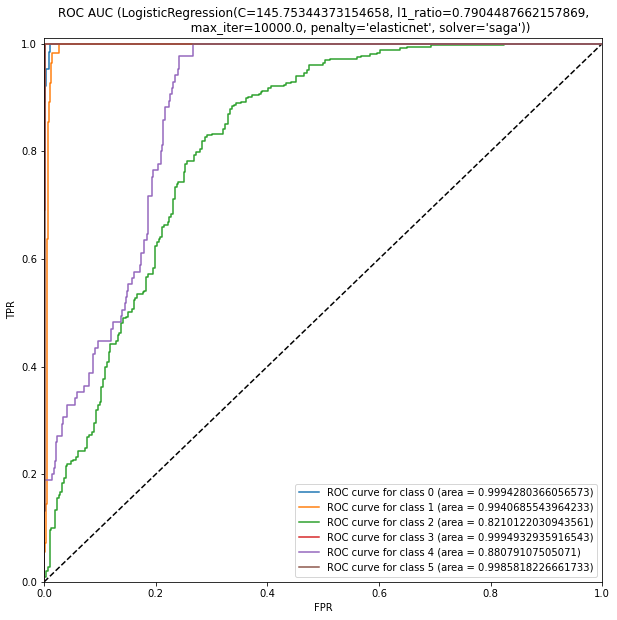
Среднее арифм-е: recall\_score = 0.9820558552741209

Средневзвешенное: recall\_score = 0.9728731816973455

Посмотрим на матрицу ошибок, полученную при обучении оптимальной модели на изначальных данных и данных после ADASYN:



На данной матрице ошибок (здесь оптимальное решение обучено на дисбалансированной выборке, а валидация проводится на сбалансированной) можно найти подтверждение проблемам дисбаланса: многие образцы ошибочно предсказываются как major-классы. Соответственно, после обучения модели на сбалансированных данных вышеотмеченная проблема исчезает. Результаты от oversampling-а превзошли все ожидания. ROC-кривая по классам:



# Машина опорных векторов

# Линейная модель

Начнём с линейной модели, *LinearSVC*. Результаты *baseline*-решения:

1. Общая точность (accuracy) = 0.8924485125858124

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.5

Класс 1: precision\_score = 0.0

Класс 2: precision\_score = 0.8926829268292683

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.92

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.3854471544715447

Средневзвешенное: precision\_score = 0.8601488265598168

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.047619047619047616

Класс 1: recall\_score = 0.0

Класс 2: recall\_score = 0.997275204359673

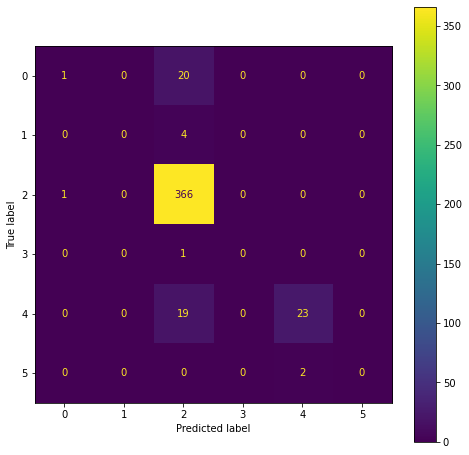
Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 0.5476190476190477

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.2654188832662947

Средневзвешенное: recall\_score = 0.8918974632521717



Результаты оптимальной модели:

1. Общая точность (accuracy) = 0.9336384439359268

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 0.0

Класс 2: precision\_score = 0.9338422391857506

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.9142857142857143

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.4746879922452441

Средневзвешенное: precision\_score = 0.921911686586571

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.42857142857142855

Класс 1: recall\_score = 0.0

Класс 2: recall\_score = 1.0

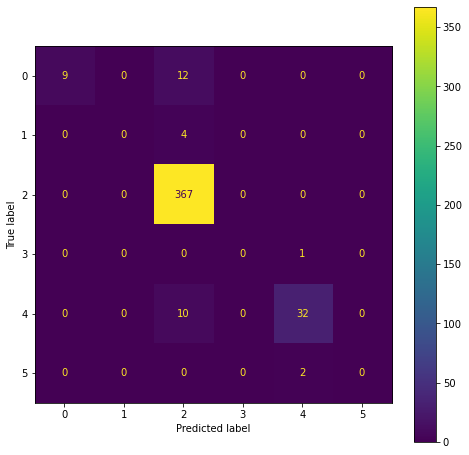
Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 0.7619047619047619

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.3650793650793651

Средневзвешенное: recall\_score = 0.93340965464606



Теперь по очереди используем методы восстановления баланса выборки:

1. RUS (*Random UnderSampling*):

1. Общая точность (accuracy) = 0.8808290155440415

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.9523809523809523

Класс 1: precision\_score = 0.0

Класс 2: precision\_score = 0.8872180451127819

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.8205128205128205

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.4433519696677591

Средневзвешенное: precision\_score = 0.8719054472074687

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.7142857142857143

Класс 1: recall\_score = 0.0

Класс 2: recall\_score = 0.9672131147540983

Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 0.9411764705882353

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.43711254993800797

Средневзвешенное: recall\_score = 0.9366990727339009

1. ENN (*Edited Nearest Neighbours*):

1. Общая точность (accuracy) = 0.9362880886426593

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 1.0

Класс 2: precision\_score = 0.9426751592356688

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.8648648648648649

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.6345900040167556

Средневзвешенное: precision\_score = 0.9331730966781409

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.5

Класс 1: recall\_score = 0.25

Класс 2: recall\_score = 0.9966329966329966

Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 0.8421052631578947

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.4314563766318152

Средневзвешенное: recall\_score = 0.9433337076364221

1. IHT (*Instance Hardness Threshold*):

1. Общая точность (accuracy) = 0.9411764705882353

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.9

Класс 1: precision\_score = 0.0

Класс 2: precision\_score = 0.9556451612903226

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.868421052631579

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.45401103565365025

Средневзвешенное: precision\_score = 0.9309255838045916

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.9

Класс 1: recall\_score = 0.0

Класс 2: recall\_score = 0.9957983193277311

Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 0.8918918918918919

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.46461503520327047

Средневзвешенное: recall\_score = 0.9668518667542205

1. SMOTE (*Synthetic Minority OverSampling Technique*):

1. Общая точность (accuracy) = 0.9147640791476408

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.9814814814814815

Класс 1: precision\_score = 0.9152542372881356

Класс 2: precision\_score = 0.8766066838046273

Класс 3: precision\_score = 0.9830508474576272

Класс 4: precision\_score = 1.0

Класс 5: precision\_score = 0.9833333333333333

Среднее арифм-е: precision\_score = 0.9566210972275342

Средневзвешенное: precision\_score = 0.893935471916861

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.7910447761194029

Класс 1: recall\_score = 1.0

Класс 2: recall\_score = 0.9884057971014493

Класс 3: recall\_score = 1.0

Класс 4: recall\_score = 0.4864864864864865

Класс 5: recall\_score = 1.0

Среднее арифм-е: recall\_score = 0.8776561766178897

Средневзвешенное: recall\_score = 0.9349248663512814

1. ADASYN (*Adaptive Synthenic Algorithm*):

1. Общая точность (accuracy) = 0.9172932330827067

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 0.9137931034482759

Класс 2: precision\_score = 0.8920308483290489

Класс 3: precision\_score = 0.9649122807017544

Класс 4: precision\_score = 0.9482758620689655

Класс 5: precision\_score = 0.9464285714285714

Среднее арифм-е: precision\_score = 0.9442401109961026

Средневзвешенное: precision\_score = 0.9033830955078896

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.734375

Класс 1: recall\_score = 0.9636363636363636

Класс 2: recall\_score = 0.9830028328611898

Класс 3: recall\_score = 1.0

Класс 4: recall\_score = 0.6470588235294118

Класс 5: recall\_score = 1.0

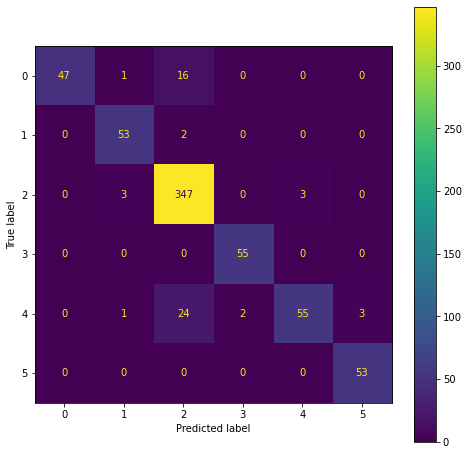
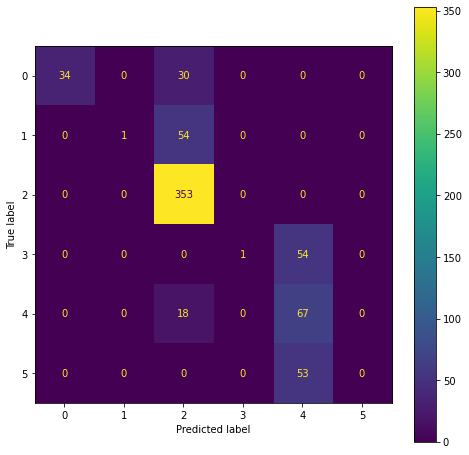
Среднее арифм-е: recall\_score = 0.8880121700044943

Средневзвешенное: recall\_score = 0.9406496022323455

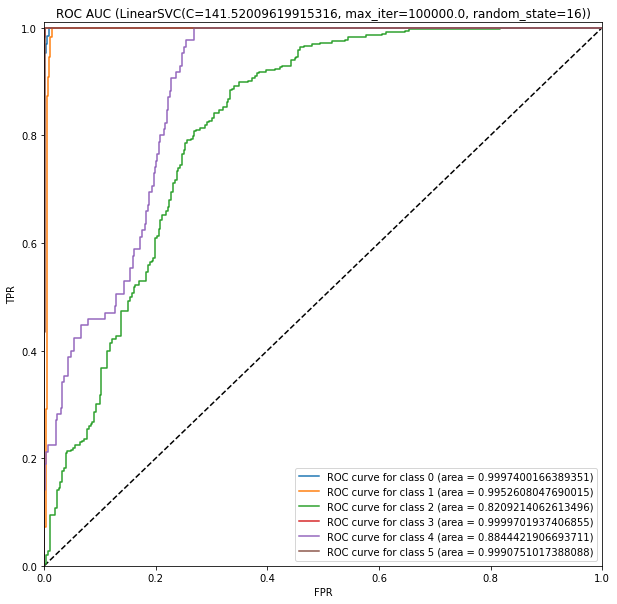
Показатели качества повторяют ситуацию с логистической регрессией: методы undersampling-а не имеют однозначного влияния на метрики (где-то станет лучше, где-то хуже - и всё это незначительно с точки зрения статистики), в то время как oversampling даёт понять, что модель "успешно" справляется со всеми классами. Условность успеха объясняется следующим:

1. Так или иначе, мы добавляем данные недостающих классов исключительно на основе уже имеющихся знаний, поэтому мы просто даём больше образцов для разгадки ("закрепляем успех"). С другой стороны, если знать что-то о предметно области, то это необязательно плохо;
2. Вдобавок к искусственному характеру данных прибавляются катастрофически малые количества образцов некоторых классов. Ни одна статистика не способна дать что-то устойчивое на выборке объёмом 5 или 7 (да и даже 14), поэтому скорее всего новые данные формируются слишком осторожно и в целом они далеки от истины;
3. Можно также заметить одну мелкую деталь: показатели качества снижаются для major-класса.

Также посмотрим на сравнение матриц ошибок для моделей, проверяемых на данных после ADASYN:



И на ROC-кривые:



# Полиномиальная модель

Переходим к стандартному классу SVC, выставляем полиномиальное ядро. Базовое решение:

1. Общая точность (accuracy) = 0.9382151029748284

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.7

Класс 1: precision\_score = 0.0

Класс 2: precision\_score = 0.9622641509433962

Класс 3: precision\_score = 0.0

Класс 4: precision\_score = 0.8666666666666667

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.4214884696016772

Средневзвешенное: precision\_score = 0.9253779165001639

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.6666666666666666

Класс 1: recall\_score = 0.0

Класс 2: recall\_score = 0.9727520435967303

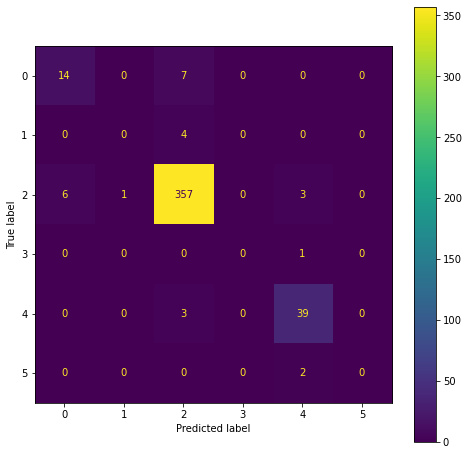
Класс 3: recall\_score = 0.0

Класс 4: recall\_score = 0.9285714285714286

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.4279983564724709

Средневзвешенное: recall\_score = 0.9376869829455735



Оптимальная модель:

1. Общая точность (accuracy) = 1.0

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 1.0

Класс 2: precision\_score = 1.0

Класс 3: precision\_score = 1.0

Класс 4: precision\_score = 1.0

Класс 5: precision\_score = 1.0

Среднее арифм-е: precision\_score = 1.0

Средневзвешенное: precision\_score = 1.0

3. Полнота (recall) по классам:

Класс 0: recall\_score = 1.0

Класс 1: recall\_score = 1.0

Класс 2: recall\_score = 1.0

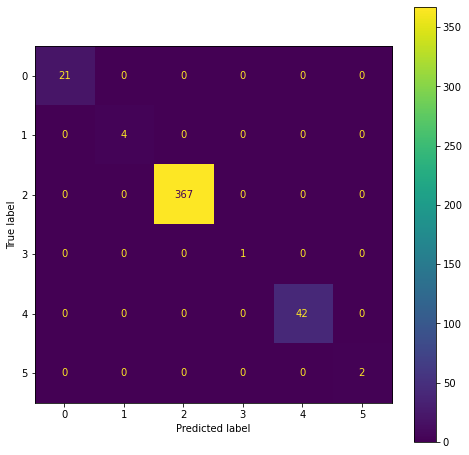
Класс 3: recall\_score = 1.0

Класс 4: recall\_score = 1.0

Класс 5: recall\_score = 1.0

Среднее арифм-е: recall\_score = 1.0

Средневзвешенное: recall\_score = 1.0



Такой подозрительно идеальный результат подсказывает, что можно перестать проверять все методы балансирования. Однако проверим для ADASYN как для самого многообещающего из них. Такая проверка необходима, поскольку единицы в метриках могут быть связаны с недостатком образцов и совершенно необязательно, что такой результат устойчив с точки зрения статистики.

1. Общая точность (accuracy) = 0.9849624060150376

2. Меткость (precision) по классам:

Класс 0: precision\_score = 0.9682539682539683

Класс 1: precision\_score = 0.9821428571428571

Класс 2: precision\_score = 0.9886685552407932

Класс 3: precision\_score = 0.9821428571428571

Класс 4: precision\_score = 0.9879518072289156

Класс 5: precision\_score = 0.9814814814814815

Среднее арифм-е: precision\_score = 0.9817735877484789

Средневзвешенное: precision\_score = 0.9873866028461978

3. Полнота (recall) по классам:

Класс 0: recall\_score = 0.953125

Класс 1: recall\_score = 1.0

Класс 2: recall\_score = 0.9886685552407932

Класс 3: recall\_score = 1.0

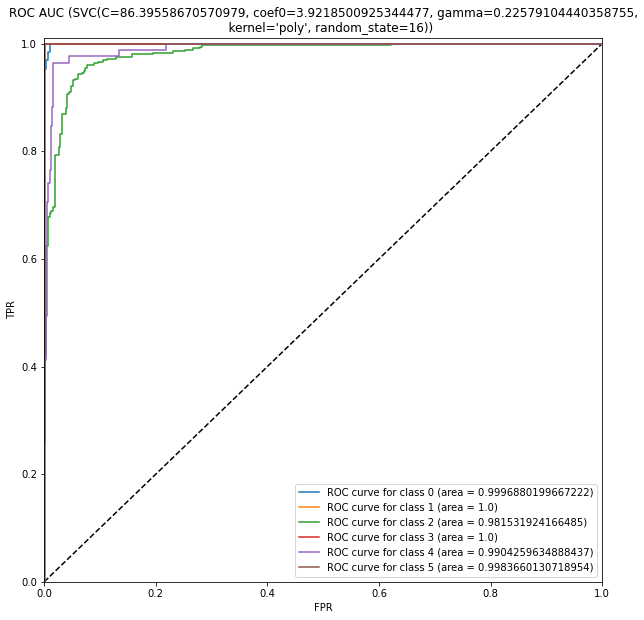
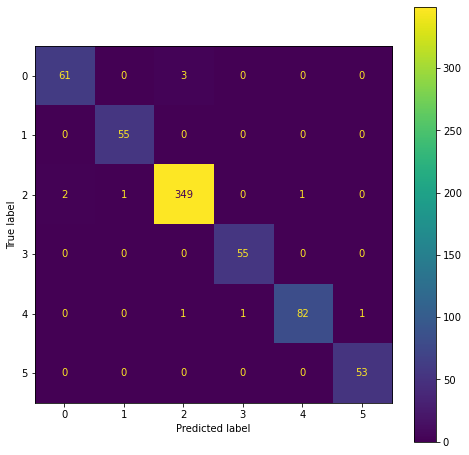
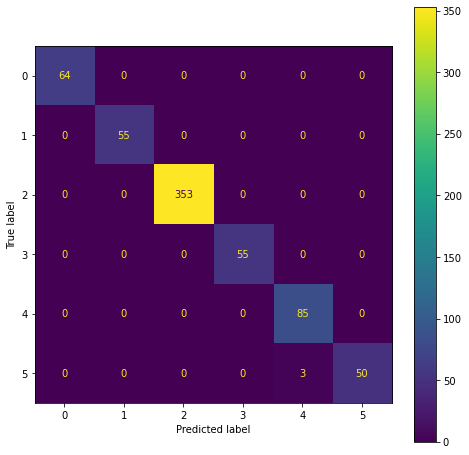
Класс 4: recall\_score = 0.9647058823529412

Класс 5: recall\_score = 1.0

Среднее арифм-е: recall\_score = 0.9844165729322891

Средневзвешенное: recall\_score = 0.9848402770497992

Снова приведём две матрицы ошибок, где модели обучены на разных данных, и ROC-кривые:



# Дерево решений

Действуем точно так же. Базовое решение:

1. Общая точность (accuracy) = 0.9954233409610984

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 1.0

Класс 2: precision\_score = 1.0

Класс 3: precision\_score = 1.0

Класс 4: precision\_score = 0.9545454545454546

Класс 5: precision\_score = 0.0

Среднее арифм-е: precision\_score = 0.8257575757575758

Средневзвешенное: precision\_score = 0.993261175001301

3. Полнота (recall) по классам:

Класс 0: recall\_score = 1.0

Класс 1: recall\_score = 1.0

Класс 2: recall\_score = 1.0

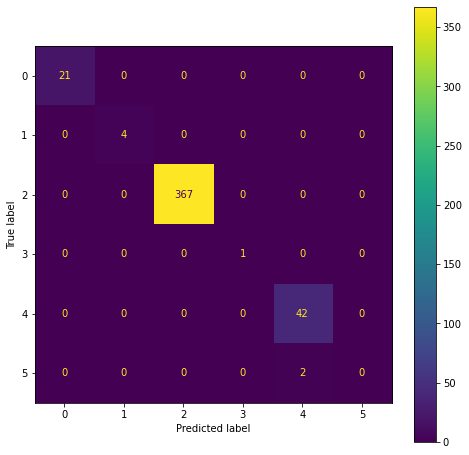
Класс 3: recall\_score = 1.0

Класс 4: recall\_score = 1.0

Класс 5: recall\_score = 0.0

Среднее арифм-е: recall\_score = 0.8333333333333334

Средневзвешенное: recall\_score = 0.9971379507727534



Оптимальное решение:

1. Общая точность (accuracy) = 1.0

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 1.0

Класс 2: precision\_score = 1.0

Класс 3: precision\_score = 1.0

Класс 4: precision\_score = 1.0

Класс 5: precision\_score = 1.0

Среднее арифм-е: precision\_score = 1.0

Средневзвешенное: precision\_score = 1.0

3. Полнота (recall) по классам:

Класс 0: recall\_score = 1.0

Класс 1: recall\_score = 1.0

Класс 2: recall\_score = 1.0

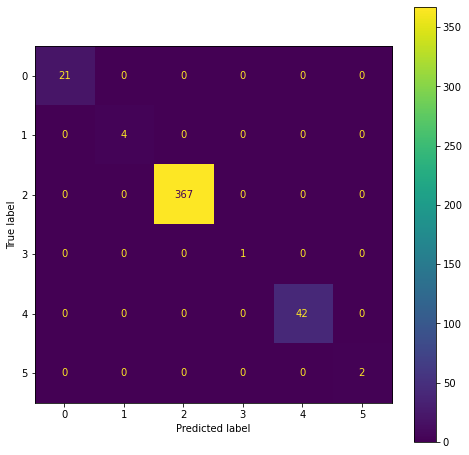
Класс 3: recall\_score = 1.0

Класс 4: recall\_score = 1.0

Класс 5: recall\_score = 1.0

Среднее арифм-е: recall\_score = 1.0

Средневзвешенное: recall\_score = 1.0



Проверка на устойчивость с помощью ADASYN:

1. Общая точность (accuracy) = 0.9924812030075187

2. Меткость (precision) по классам:

Класс 0: precision\_score = 1.0

Класс 1: precision\_score = 1.0

Класс 2: precision\_score = 1.0

Класс 3: precision\_score = 1.0

Класс 4: precision\_score = 0.9761904761904762

Класс 5: precision\_score = 0.9444444444444444

Среднее арифм-е: precision\_score = 0.9867724867724869

Средневзвешенное: precision\_score = 0.9978103051943923

3. Полнота (recall) по классам:

Класс 0: recall\_score = 1.0

Класс 1: recall\_score = 1.0

Класс 2: recall\_score = 1.0

Класс 3: recall\_score = 1.0

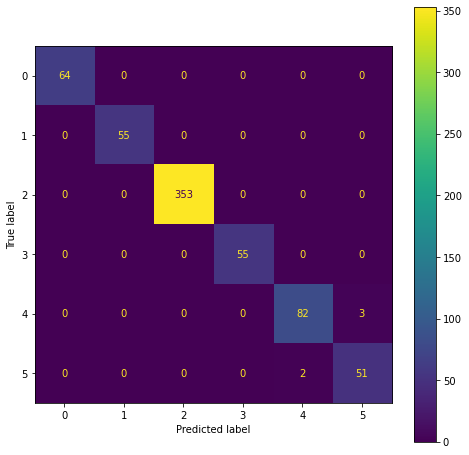
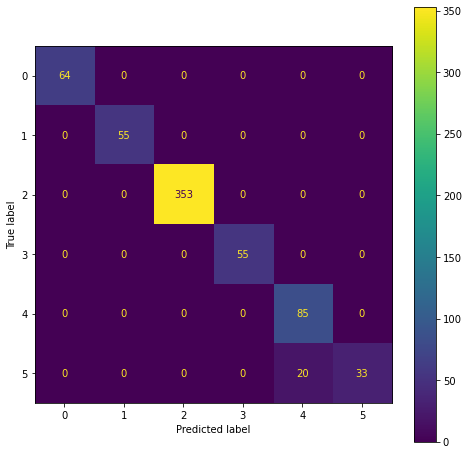
Класс 4: recall\_score = 0.9647058823529412

Класс 5: recall\_score = 0.9622641509433962

Среднее арифм-е: recall\_score = 0.9878283388827228

Средневзвешенное: recall\_score = 0.9968817957786522

Сравнение матриц ошибок:



Переход ко второму набору данных произошёл именно на этом моменте. Модель предсказывает все данные практически идеально и никакое переобучение здесь не светит. В связи с этим ансамблевые модели проверялись не здесь.

# Случайный лес

Критерии оценивания будут выглядеть слегка иначе: из-за того, что на данной выборке проблемы с классом «1», метрики *Precision*, *Recall* и *F1-score* выводятся именно для этого класса.

Здесь сразу ищем оптимальное решение с помощью случайного поиска:

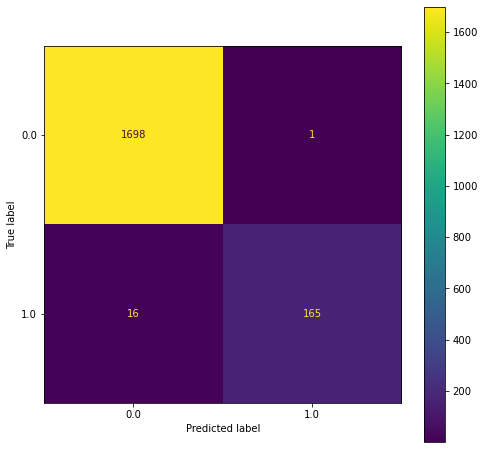
-Погрешность (accuracy, balanced) = 0.9555068142131056;

-Меткость (precision, класс 1) = 0.9939759036144579;

-Полнота (recall, класс 1) = 0.9116022099447514;

-F1 (класс 1) = 0.9510086455331411;

-ROC AUC = 0.9555068142131056.



1. IHT

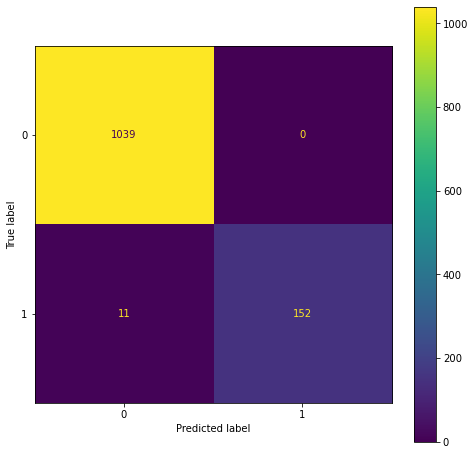
-Погрешность (accuracy, balanced) = 0.9662576687116564;

-Меткость (precision, класс 1) = 1.0;

-Полнота (recall, класс 1) = 0.9325153374233128;

-F1 (класс 1) = 0.9650793650793651;

-ROC AUC = 0.9662576687116564.



1. ADASYN

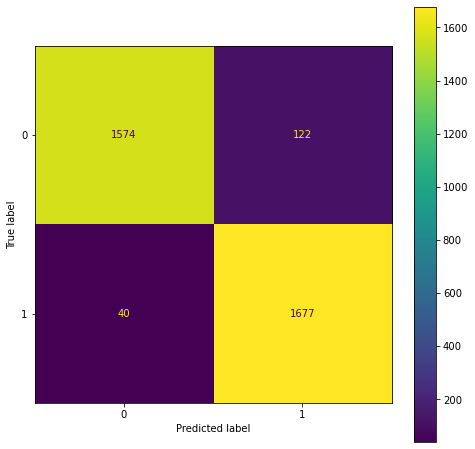
-Погрешность (accuracy, balanced) = 0.9523847952220306;

-Меткость (precision, класс 1) = 0.9321845469705392;

-Полнота (recall, класс 1) = 0.976703552708212;

-F1 (класс 1) = 0.9539249146757679;

-ROC AUC = 0.9523847952220306.



Видим, что, несмотря на все ожидания, видим прирост качества от *Oversampling*-а: повысился *Recall*.

# Градиентный бустинг

Используем библиотеку *CatBoost*. Здесь тоже сразу переходим к оптимальному решению:

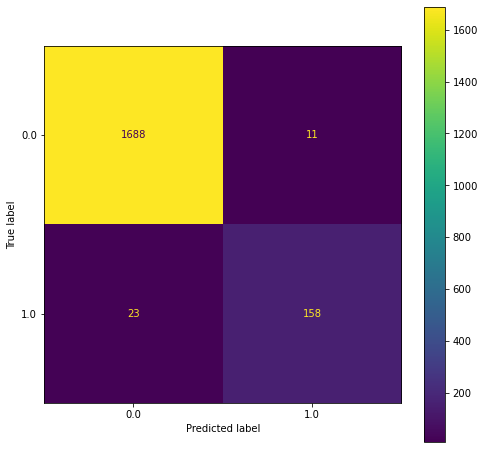
-Погрешность (accuracy, balanced) = 0.9332268900458183;

-Меткость (precision, класс 1) = 0.9349112426035503;

-Полнота (recall, класс 1) = 0.8729281767955801;

-F1 (класс 1) = 0.9028571428571428;

-ROC AUC = 0.9332268900458183.



1. IHT

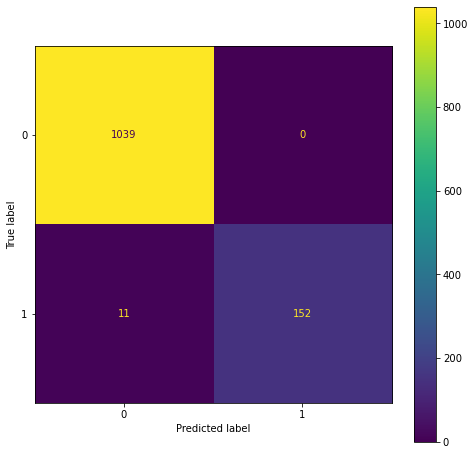
-Погрешность (accuracy, balanced) = 0.9662576687116564;

-Меткость (precision, класс 1) = 1.0;

-Полнота (recall, класс 1) = 0.9325153374233128;

-F1 (класс 1) = 0.9650793650793651;

-ROC AUC = 0.9662576687116564.



1. ADASYN

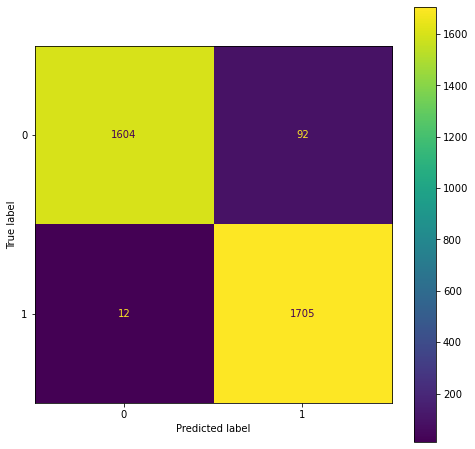
-Погрешность (accuracy, balanced) = 0.9693828913967979;

-Меткость (precision, класс 1) = 0.9488035614913745;

-Полнота (recall, класс 1) = 0.9930110658124636;

-F1 (класс 1) = 0.9704040978941376;

-ROC AUC = 0.9693828913967979.



# Заключение

В ходе выполнения курсовой работы было изучено влияние методов балансирования выборки на качество модели, а также закреплён полученный в ходе лабораторный навык работы с моделями машинного обучения. Дать однозначный вывод по важности баланса классов (даже ценой искусственных данных) можно не везде, поскольку видели, что для ансамблевых моделей метод ADASYN повысил качество модели в той ситуации, когда не должен был. С уверенностью можно заявить о том, что для датасета с орбитами ожидаемо помогают все методы *Oversampling*-а, а для датасета с пульсарами – *Undersampling*-а. Открытым остаётся вопрос о том, какое количество искусственных данных является достаточным для того, чтобы модель машинного обучения справилась с задачей на достойном уровне.

# Список использованных источников

1. Курс лекций по технологиям машинного обучения // YouTube URL : <https://youtube.com/playlist?list=PL9vFTJYocFHoRvWr-jiC9r3i9jayLzvPg>;
2. Курс лекций по методам машинного обучения // YouTube URL : <https://youtube.com/playlist?list=PL9vFTJYocFHqnXejx0hpRKxVEH0fny6Ls>;
3. Документация библиотеки *scikit-learn* // URL : <https://scikit-learn.org/stable/index.html>;
4. Документация библиотеки *imbalanced-learn* // URL: <https://imbalanced-learn.org/stable/references/index.html>;
5. Документация библиотеки *SciPy //* URL: <https://www.scipy.org/>;
6. «*Нужно ли бояться несбалансированности классов?*» // Хабр URL : <https://habr.com/ru/post/349078/>;