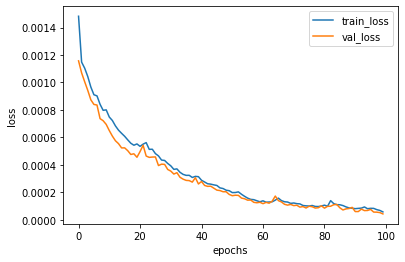
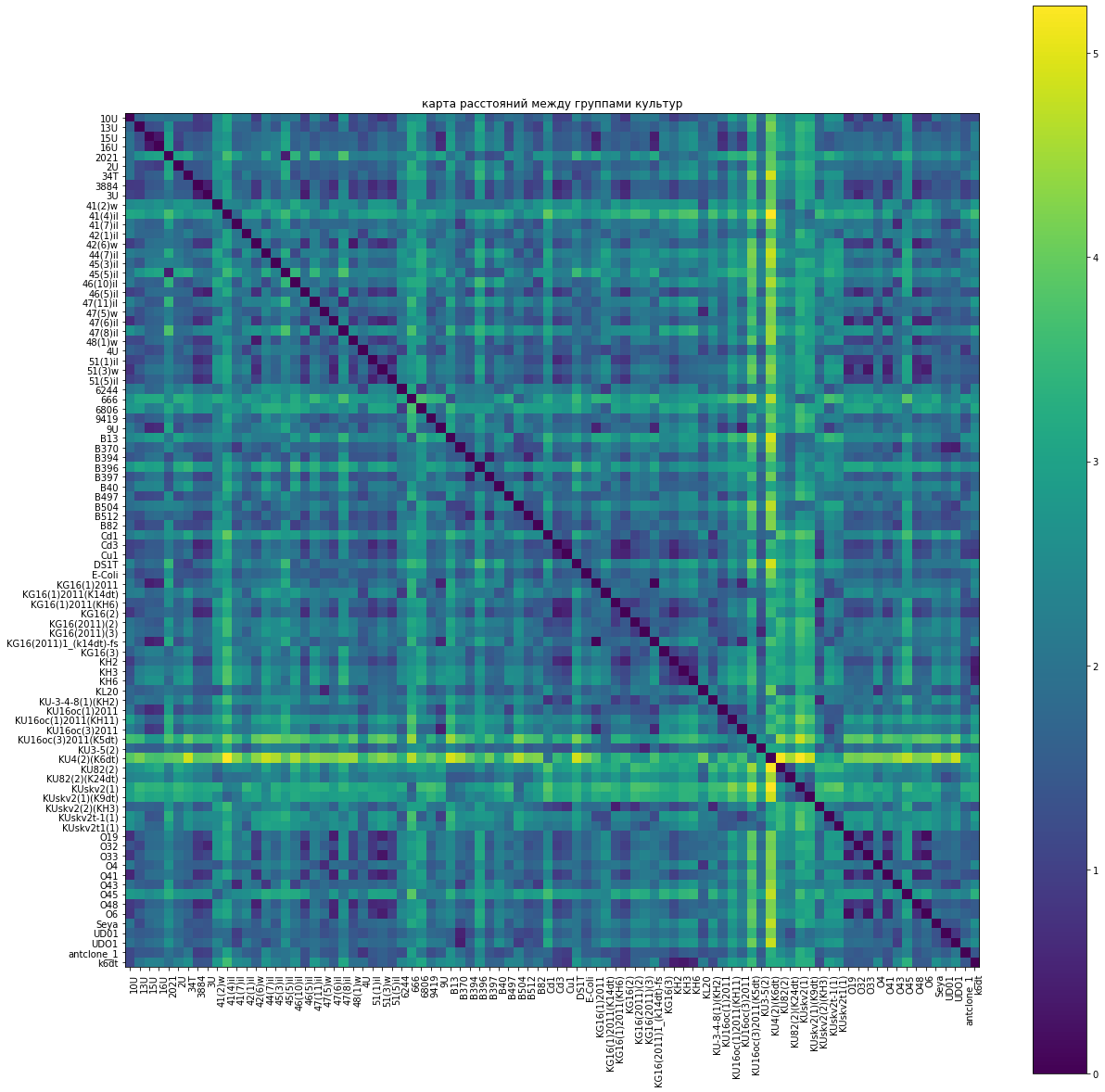
**DAE**

Сегодня при тренировке у DAE при использовании датасэта, который я строила прибавлением вектора, у которого координаты взяты из нормального распределения (если не понятно, вот, что написано про этот метод (np.random.normal() из свободной библиотеки numpy) в документации: “Draw random samples from a normal (Gaussian) distribution.”) резко просела точность при классификации (что бы я не пробовала, она всегда была в районе 40%). Это наблюдалось и при использовании старой модели, которая на этих данных выдавала 94%. Не знаю с чем это связано, random.seed такой же, не помогает ни увеличение деревьев в лесу, ни увеличение размерности скрытого пространства. Поэтому я решила снова пересмотреть метод шума. Я посмотрела, что максимальная координата у случайного вектора с нормальным распределением ≈ 4 (потому что я не правильно перевела scale из документации, думала, что это ширина в смысле значения, а они имели ввиду дисперсию), а максимальная интенсивность в профилях – единица, то есть шум это примерно 400% от данных, конечно классификация не будет работать (вообще не знаю как у меня получались при такой аугментации 94%). Поэтому я решила нормировать случайный вектор и дополнительно разделить его на 10 (то есть взять на шум ≈ 10% от данных), таким образом, это отличается от прошлого шума только параметром noise\_factor = 1/40 (уточню noise\_factor – просто параметр шума, в данном случае он фиксированный). При таком раскладе получается следующий график лоссов:



Heat map (метод построения тот же):



И classifier report (лес учила на 70% от 3170 профилей с шумом аналогичным шуму в обучении DAE(см. выше)):

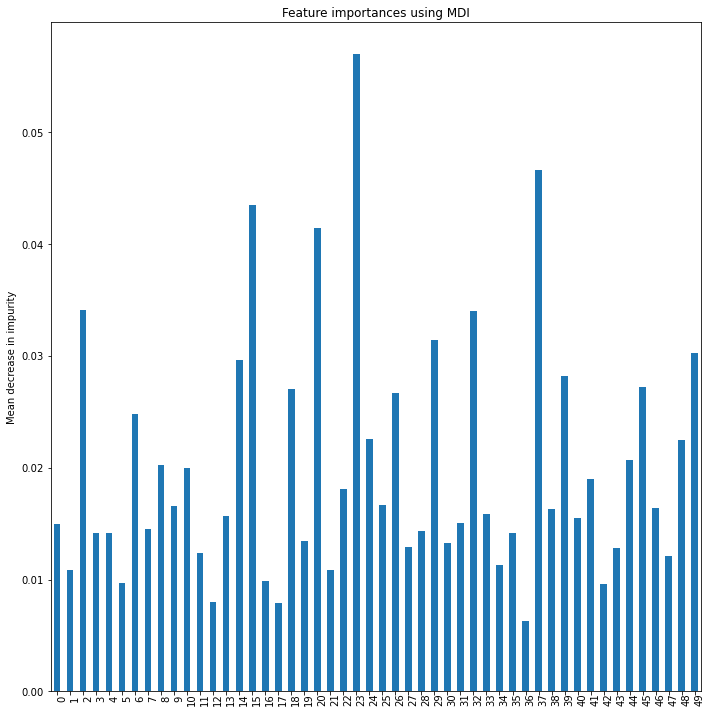


Про формулы для TP TN FP FN: что значит формулы для них? У них просто определения есть и все. Можно там картинку нарисовать их определения через confusion matrix для мультиклассовой классификации.

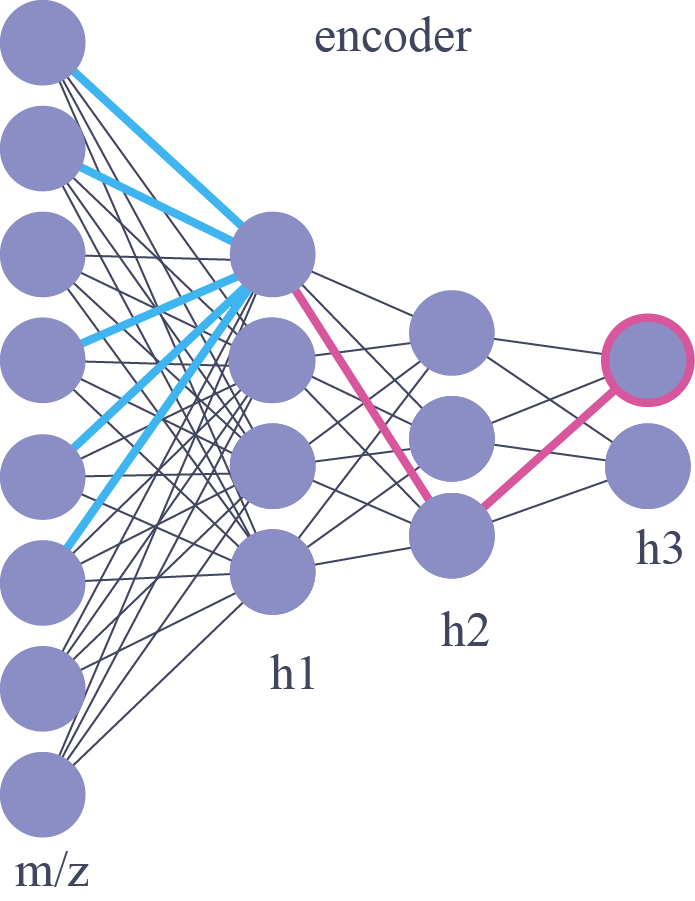
Про ROC-кривую: я почитала, она строится только для бинарной классификации, для мультиклассовой аналога ей нет. Так что эта табличка пока все что я могу предложить для оценки точности классификации.

**Лес**

Использовался RandomForestClassifier из свободной библиотеки sklearn со стандартными параметрами (число деревьев = 100, критерий деления – gini, минимальное для деления количество элементов в листе = 2, максимальное количество features рассматриваемых для лучшего разбиения sqrt от количества features (в документации параметров намного больше, кратко описала как смогла самые важные, по моему мнению, думаю, можно будет просто сослаться на доки)). После построения на инференсе, получили таблицу показанную выше, и с помощью метода feature\_importance\_ из той же библиотеки (на сколько я поняла, он основан на поиске наибольшего прироста информации (с использованием критерия gini), если что в документации написано: “Feature importance based on mean decrease in impurity”) получили следующий график важности компонент:



Я взяла топ 10 (см выделенные нейроны в слое h\_3 на рисунке) и по ним искала важные features в исходном пространстве. Использовала метод, описанный в статье из nature, в разделе “Identification of Informative m/z peaks”, с небольшим изменением. У них в енкодере 3 слоя, а у нас 4, поэтому процедуру, которую они делали между h\_2 и h\_1 слоями, я сделала между h\_3 и h\_2, и между h\_2 и h\_1, то есть нашла максимальные веса w\_ij последовательно два раза (см, пурпурные линии на рисунке). После этого между m/z и h\_1 слоями я считала T из предложенного метода и искала во входном слое все нейроны, где вес w\_jd >= T (см. голубые линии на рисунке) , таким образом я получила массив из 4783 важных m/z features (текстовый файл с ними, на всякий случай прикреплю к письму).



(Комментарий к самому методу: во-первых, у них явно опечатка, формула такая:

И потом они ищут по d веса большие T, но, d в нашем случае меняется от 0 до 6000, а мы должны искать по входному слою, в общем, я поменяла местами d и j. Плюс я не совсем поняла смысл β, по идее, если у β – нормальное распределение, то T – случайная величина из распределения весов, но видимо они подняли мат ожидание β и сузили ей дисперсию. Я не совсем понимаю: β нужно генерировать случайно или фиксировать в зависимости от того, сколько нам надо получить фич? Я генерировала β для каждого j, но, если нам нужно меньше фич, думаю просто зафиксировать β = 2.5.)

**Сравнение со старым методом**

Я выбрала случайные 20 бактерий (по айдишникам) и создала три выборки для сравнения каждая размером 220 бактерий (csv файлы я вам прикреплю, скиньте их Ефимову, что бы он посчитал recall, precision, f1-score, accuracy). Натренированное дерево на noise norm (который новый) выдало 100% по всем параметрам, на half noise 1e-1 – точность 95%, на full noise 1e-1, ой-ой, страшные 52% точности (хотя интересно как с этим сэтом справится старый метод). Exсel таблицы для каждого датасэта c результатами классификации (как та, что выше) прикреплю к письму.

Так же прикреплю к письму все оригиналы картинок из этого отчета в хорошем качестве и первую табличку, пусть они у вас уже будут.