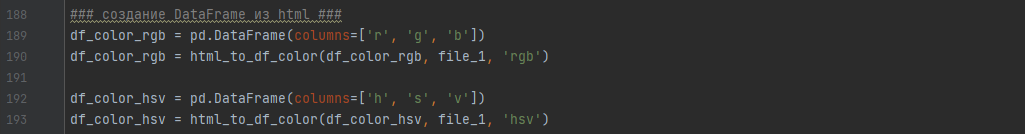
**Программирование в системах информационной безопасности  
 Лабораторный практикум**

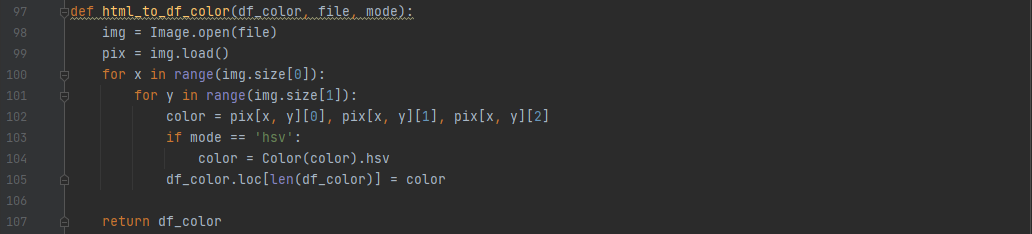
Васюткин Александр Олегович, группа 181-331

**Задание 6.**

1. Построить наивный классификатор Байеса для присвоения пикселям изображения меток «кожа» и «не кожа». За признаки пикселей следует принять характеристики цвета в цветовой модели RGB.
2. Построить наивный классификатор Байеса для присвоения пикселям изображения меток «кожа» и «не кожа». За признаки пикселей следует принять характеристики цвета в цветовой модели HSV.



Для начала преобразуем HTML-файл с изображением в DataFrame. Делаем мы это при помощи функции, входными параметрами которой являются DataFrame, в который мы будем записывать цвета, изображение и параметр цветовой модели.

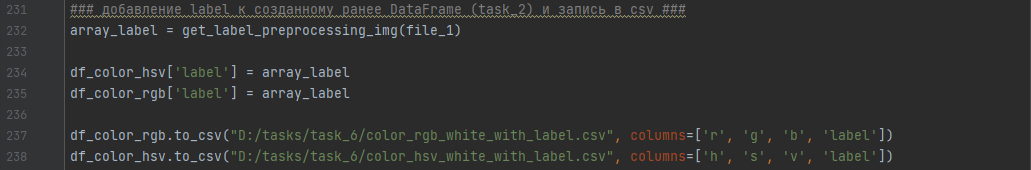


Открываем и поочередно обрабатываем каждый пиксель переданного изображения, и преобразовываем RGB в HSV в зависимости от переданного параметра. Записываем преобразованные цвета в DataFrame.

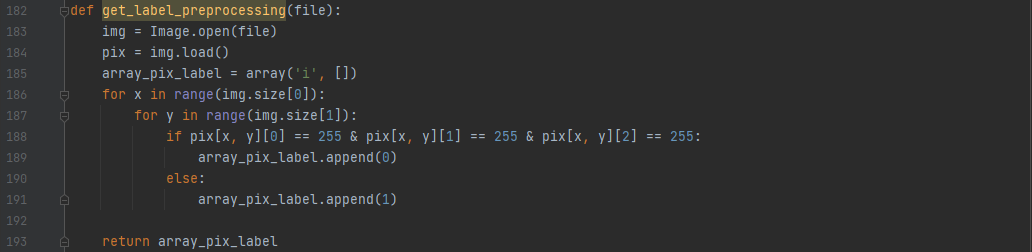
Были созданы четыре DataFrame, содержащие цвета RGB и HSV для следующих изображений (два для обычных изображений и два для изображений, где все, кроме кожи закрашено белым):



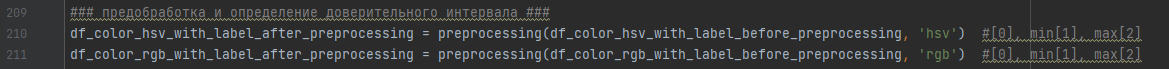




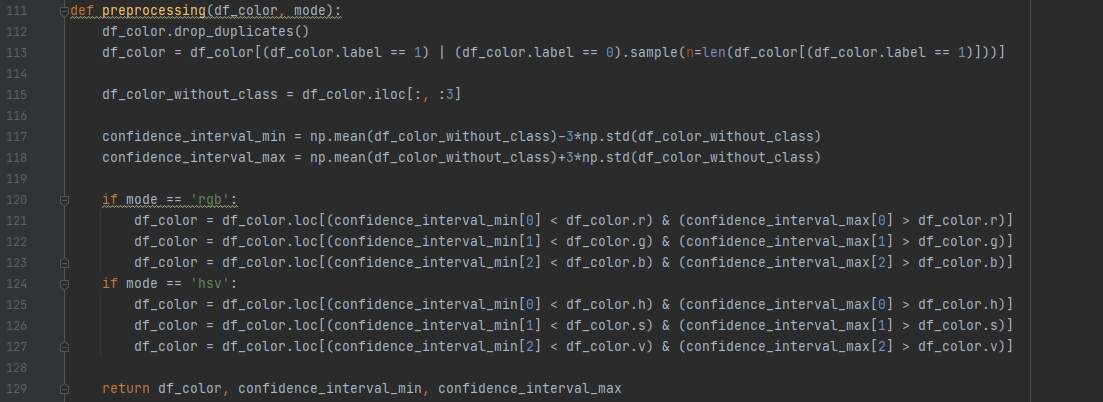
Затем нам необходимо найти “кожу” на изображении. Делаем мы это при помощи функции, которой на вход подаем DataFrame, содержащий цвета, и изображение, где все участки кроме кожи, заменены белым цветом, собственно, на котором мы и будем искать области с “кожей”. Добавляем полученный массив из нулей и единиц к нашему DataFrame (для обычных изображений), он будет служить меткой. После завершения работы функции мы сохраняем полученные DataFrame в файл формата csv.



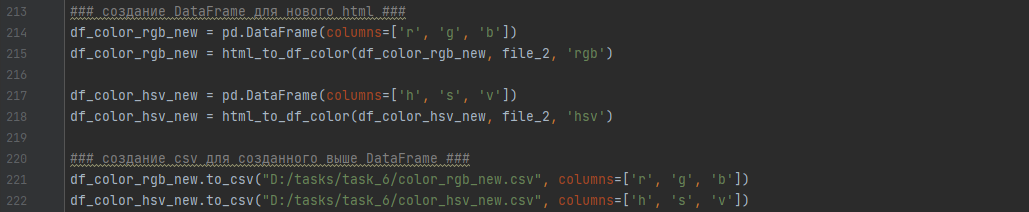
Поочерёдно обрабатываем каждый пиксель изображения и заполняем массив нулями, если цвет белый, и единицами – если не белый (кожа). Функция возвращает массив с метками.



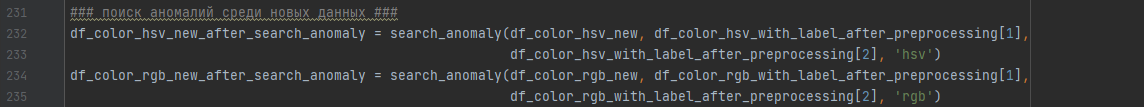
Теперь нам необходимо предобработать наши данные и вычислить доверительный интервал. Входными данными функции является DataFrame, который будем предобрабатывать, и параметр цветовой модели.



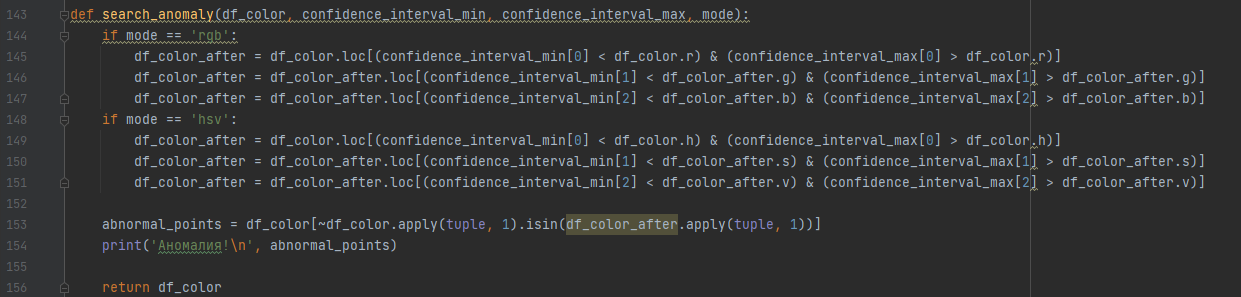
Уравниваем количество представителей классов и удаляем дубликаты. Ищем левую и правую границы доверительного интервала и отсеиваем объекты, которые не входят в них. Функция возвращает DataFrame, содержащий только те объекты, которые входят в доверительный интервал, и левую и правую границы доверительного интервала.



Создаем новый DataFrame, в который помещаем цвета, которые будем классифицировать, из HTML-файла. И записываем его в csv файл.

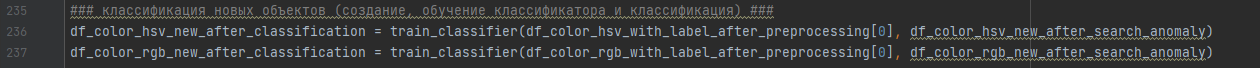


Найдем аномальные объекты, которые не входят в доверительный интервал, в новых данных. На вход функция получает DataFrame, в котором нужно искать аномалии, левую и правую границы доверительного интервала и параметр цветовой модели.

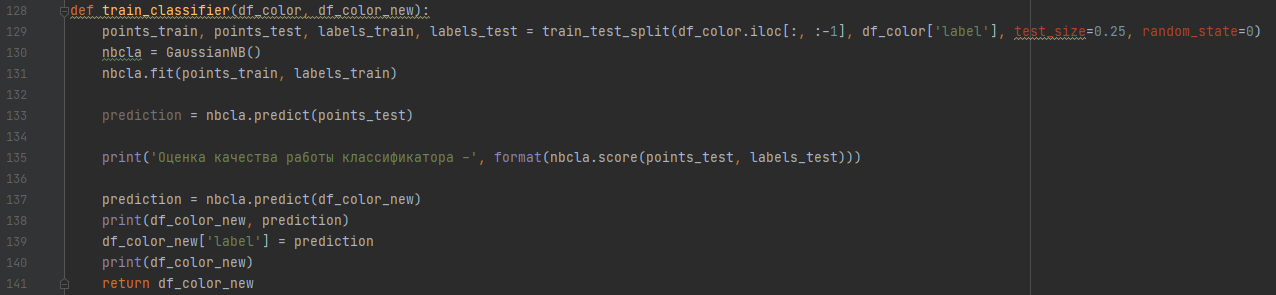


Просеиваем новые объекты через доверительный интервал. Далее, чтобы выявить аномальные объекты, мы находим строки, которые не прошли просеивание через доверительный интервал, путем нахождения различий между исходным сетом со всеми новыми данными и новыми данными, входящими в доверительный интервал. Выявленные аномальные объекты:





Далее нам необходимо классифицировать новые объекты. На вход функции подаем DataFrame объектами, прошедшими предобработку, которыми будем обучать классификатор и DataFrame с новыми объектами, которые будем классифицировать.



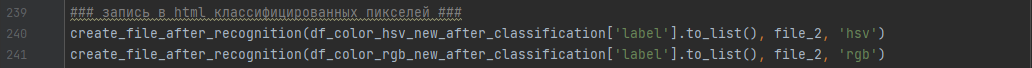
Делим данные на тренировочные и тестовые, отделяя от столбца с метками. Создаем и обучаем классификатор на тренировочных данных, проверяем на тестовых. Оценка качества работы для RGB:



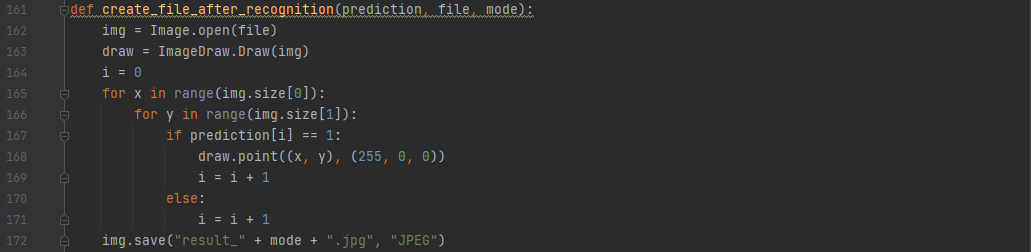
Для HSV:



Классифицируем новые объекты и добавляем к DataFrame с новыми объектами столбец, содержащий метки.



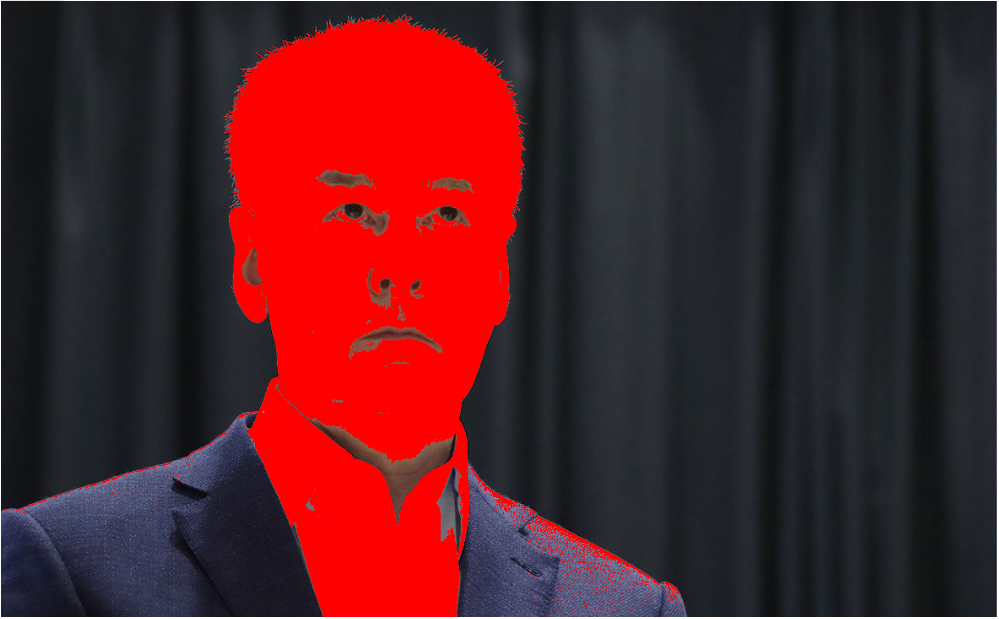
Создаем изображение, которое будет содержать классифицированные цвета. Входными параметрами функции являются массив с метками новых объектов, файл, цвета из которого мы классифицировали, и параметр цветовой модели.



Создаем файл, в который будем записывать классифицированные цвета. Открываем файл, цвета из которого мы классифицировали, и построчно переписываем его в созданный файл, опираясь на значение метки в переданном массиве. Если значение равно единице, то вместо исходного цвета записывается красный. Закрываем оба файла.

**Результат.**

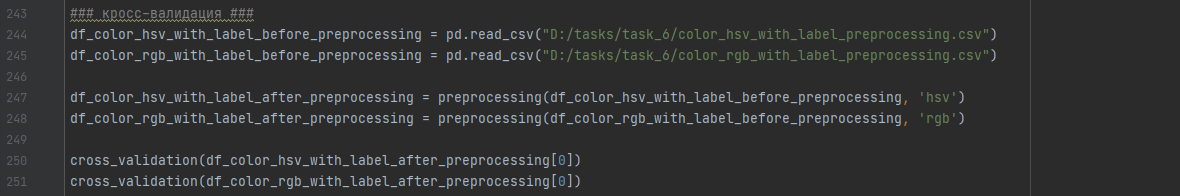
Оригиналы и файлы, полученные в результате классификации объектов, признаками которого являлись характеристики цвета в цветовой модели RGB и HSV соответственно:



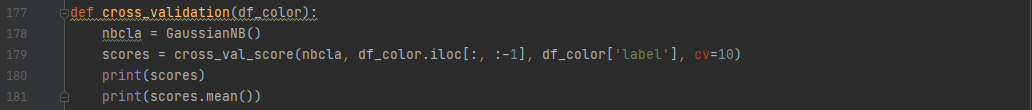




1. Провести кросс-валидацию наивного классификатора Байеса на данных из заданий 6.1 и 6.2 (независимо). Сравнить и объяснить результаты.



Считываем записанные ранее в файл DataFrame с метками, предобрабатываем их и подаем на вход функции кросс-валидации.



Создаем классификатор и вызываем функцию кросс-валидации, подавая на вход объект классификатора, DataFrame и метки по отдельности, а также указываем количество частей, на которые будем разбивать наш DataFrame. Результат работы функции кросс-валидации для HSV и RGB соответственно:

