Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Томский политехнический Университет»



Институт

ИЯТШ

01.03.02 «Прикладная математика и информатика»

**Отчет о практике**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Выполнил:** |  | | | | |
| студент группы | 0В91 |  | Змитрович Н.Л. |  |
|  |  |  |  |  |
| **Проверил:** | Чумаков А.Г. | | | | |
| преподаватель |  |  |  |  |  |
|  |  |  |  |  |  |

Томск – 2021

**Задание**

Используя данные и их описание с источника <https://www.kaggle.com/uciml/pima-indians-diabetes-database> сделайте анализ и обучите модель предсказывать наличие диабета.

**Ход работы**

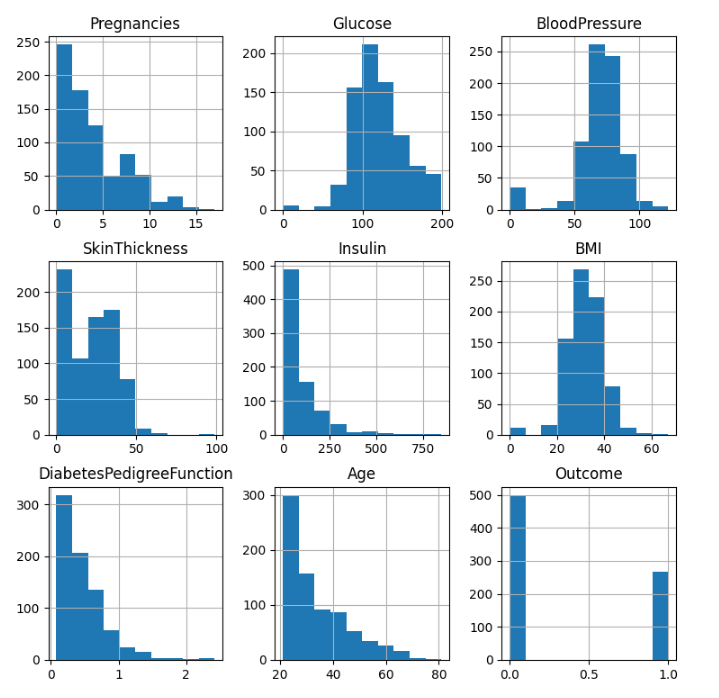
**Инициализация данных**

Для выполнения работы скачаем данные с сайта kaggle и создадим папку data в директории с jupyter notebook. Инициализируем библиотеки pandas для работы с данными и matplotlib, для построения графиков, а также применим функцию самого jupyter notebook %matplotlib notebook, для того, чтобы в дальнейшем можно было изменять размеры графиков. После чего средствами pandas импортируем базу данных и отобразим несколько первых элементов, для визуального понимания структуры данных.

**Первичный анализ**

Имеет восемь различных независимых переменных и одну зависимую – **Outcome.** Если значение Outcome пациента равно один, значит у него обнаружен диабет, если ноль, значит у пациента нет диабета. С помощью метода describe проверим основные параметры каждой из переменных, а также узнаем количество пациентов. При отображении первых нескольких строк таблицы, мы можем увидеть, что в таблице нет неопределенных значений, но для того, чтобы убедиться в этом применим комбинацию методов - isnull() и sum(). Метод isnull() вернет таблицу у которой этот метод вызван, только теперь все пустые ячейки (значение в которых равно null) будут иметь значение False, а непустые значение True. Так как в языке Python значение True при суммировании эквивалентно единице, False нулю, то после суммирования по каждой переменной, получим количество пустых ячеек в столбцах. По результату можем сделать вывод, что пустых ячеек в таблице нет.

Также при первичном ознакомлении с базой данных можно заметить что значение Insulin и SkinThicknes принимают значение 0, чего быть не может, значит значения просто пропущены. Построим гистограммы частот для каждой переменной, так как графиков много, чтобы они не наслаивались друг на друга и были легко читаемы в методе hist() передадим параметр figsize=(9,9), чтобы увеличить высоту и ширину основной фигуры в которой будут находится десять графиков. Действительно, по гистограммам частот параметров Insulin и SkinThicknes видим, что имеется достаточно большое количество нулевых значений. Попробуем посчитать это количество для каждого столбца, исключая параметры в которых нулевые значения теоретически могут присутствовать: Pregnancies (количество беременностей), Age (возраст) и Outcome (значения в котором в принципе могут быть равны либо ноль, либо один). В результате получаем, что у всех параметров, кроме DiabetesPedigreeFunction, имеются пропущенные значения, попробуем их заменить. По гистограммам частот можно сделать вывод, что параметры Glucose, BloodPressure и BMI имеют распределения похожие на нормальное, а остальные параметры, кроме Outcome, распределения похожие на параметрическое, значит заменять пропущенные значения на среднее было бы не очень корректно, так как среднее у параметрического распределение сильно смещено от максимального значения, а нам бы хотелось заменить пустые значения на такие, которые имеются у большинства, но при этом опять же из-за того, что некоторые параметры распределены экспоненциально, заменять пропущенные значения на моду, было бы так-же не совсем корректно, поэтому заменим их на медиану. Такая замена, как мне кажется не окажет критического влияния на обучение модели, и не вызовет переобучение на так называемых шумах, которыми по сути и будут являться пустые значения. Сделаем замечание, у параметра SkinThicknes и Insulin, количество пропущенных значений крайне велико относительно всех значений (30% для SkinThicknes и 49% для Insulin).



(Рис 1. Гистограммы распределения параметров)

**Подготовка данных**

Перед тем, как производить замену нулевых значений, сохраним индексы элементов с ненулевыми Insulin SkinThicknes. Теперь найдем медианные значения для каждого из параметров, который будем изменять. Важное замечание, при поиске медианы важно исключить пустые значения, так как они могут сильно сместить положение медианы в распределении. Так, например, для инсулина, если не исключать пропущенные значения, медиана будет равна 30.5, а если исключать, то уже 125. Теперь создадим список с «фичами» нашей базы данных и пройдемся по каждому столбцу, кроме Pregnancies, Age и Outcome (первого и двух последних), заменяем нулевые значения на значение медианы. Медиану будем искать на каждой итерации цикла с помощью маски, которая исключает нулевые значения, после чего применим метод median() и возьмем только текущий столбец. Проверим результат и увидим, что теперь пропущенных (нулевых) значений в таблице нет.

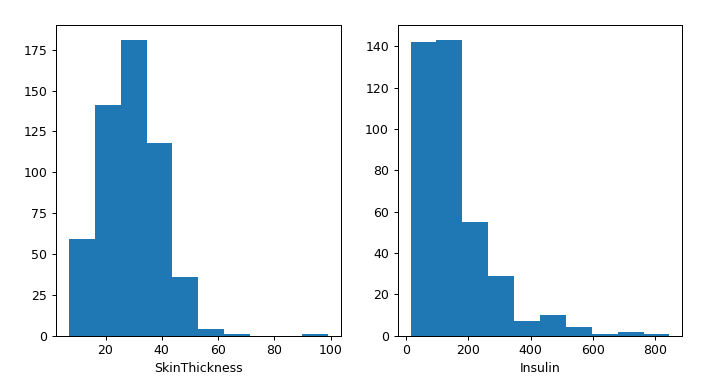
С помощью метода corr() посмотрим на значения корреляции между каждым из параметров. Некоторые параметры имеют достаточно большую корреляцию, например, значение корреляции между возрастом и количеством беременностей достаточно высока (0.544) или глюкозой и инсулином (0.419), значит не стоит использовать регрессионную модель, так как в такой модели нежелательно иметь высокую корреляцию между независимыми переменными. Также заметим, что корреляции SkinThicknes Outcome и Insulin Outcome, достаточно небольшие, а учитывая, что почти треть стобца SkinThicknes и половина столбца Insulin были пропущены, можно считать эти столбцы недостаточно значимыми, для обучения модели и просто не включать в обучающие данные.

Попробуем посмотреть корреляцию между SkinThicknes, Insulin и Outcome в изначальной базе данных, при этом исключая нулевые значения. Для этого воспользуемся двумя, сохраненными до этого, списками, с элементами в которых не пропущены SkinThicknes и Insulin, Получаем значения корреляции между SkinThicknes и Outcome равное 0.303 и значения корреляции между Insulin и Outcome равное 0.258. Учитывая, что корреляция между SkinThicknes и Outcome, и Insulin и Outcome, в таблице с заменой пропущенных значений соответственно равны 0.215 и 0.204. То есть добавив в данные 30% у SkinThicknes и почти 50% у Insulin медианных значений, мы достаточно сильно ухудшаем корреляцию.

Также учитывать специфику обучения модели методом решающих деревьев. Может получиться так, что пустые значения, которые мы заменили на медиану будут играть большое значение при обучении, так как дерево определяет некоторый барьер, и если значение меньше барьера, то оно будет определено в один лист, а если больше, то в другой. Таким образом мы рискуем определить значительное количество элементов в неправильный лист (так как изначально, значение было пропущено). Поэтому мне кажется, из базы данных стоит вычленить некоторых валидационный датасет, на котором можно проверить будут ли лучше метрики, если обучать модель на данных с заменой пропущенных значений, с заменой пропущенных значений и удалением параметров SkinThicknes и Insulin и с использованием базы данных с исключением пропущенных значений в этих колоннах (только в этих, потому что в остальных количество пропущенных значений мало).

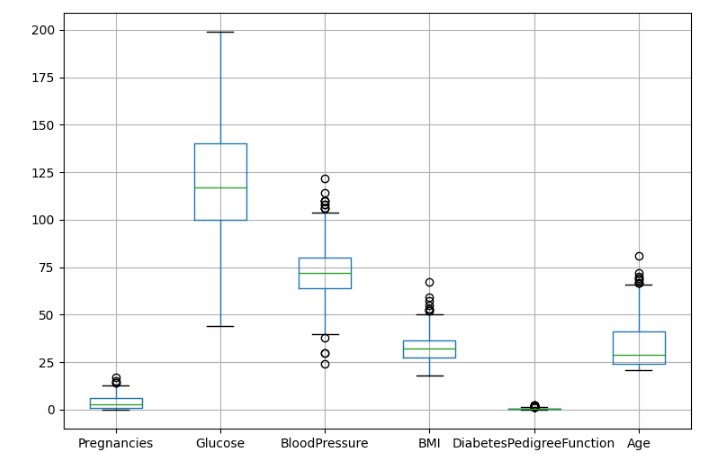
Помимо пропущенных значений в данных могут присутствовать так называемые выбросы. Если взять расстояние между первых и третьим квартилем выборки и отложить от этих квартилей отрезки равные этому расстоянию, умноженному на полтора, то все значения, которые не входят такой интервал называются выбросами, которые стоит исключать при обучении, так как модель может переобучиться на этих выбросах. При замене пустых значений на медианные мы изменяем саму первоначальную функцию распределения, изменяя стандартное отклонение и в некоторых случаях среднее значение. Но так-как во всех параметрах кроме толщины кожи и инсулина количество пропущенных значений мало относительно всех значений, пустые измененные значения можно не отбрасывать при поисках выбросов.

Посмотрим на графики гистограмм частот, и заметим, что скорее всего в каждом параметре имеются выбросы. Также для параметра Insulin и SkinThickness, стоит перестроить гистограммы без учета изменённых значения, которые первоначально были нулевыми. Используем массивы индексов исключающие пустые значение, которые мы сохранили до этого и построим гистограммы для двух параметров (рис. 2). Заметим, что данные распределения тоже имеют выбросы.



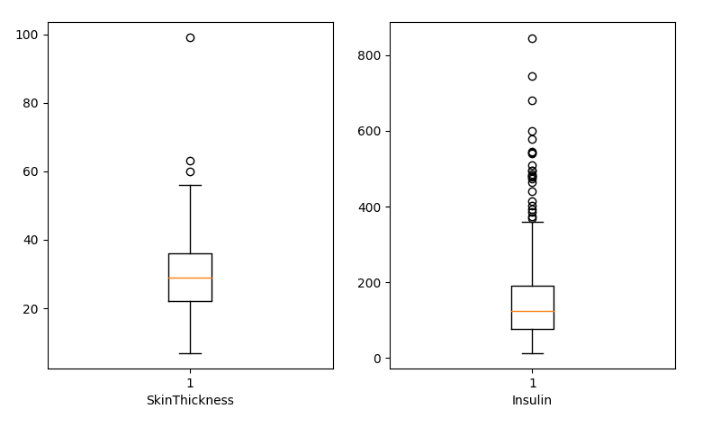
(Рис.2 Графики гистограмм частот без учета пустых значений)

Для того, чтобы точно убедиться в наличии выбросов построим диаграммы box plot. Причем для всех параметров, кроме SkinThickness и Insulin будем учитывать все значения, включая и измененные, так как их количество мало по сравнению с общим количеством значений.



(Рис.3 Графики box-plot для параметров в которых учитываем измененные значения)

Заметим, то практически во всех параметрах имеются выбросы. Теперь воспользуемся массивами индексов непустых изначально значений и построим графики box-plot отдельно для параметров SkinThickness и Insulin (рис.4).



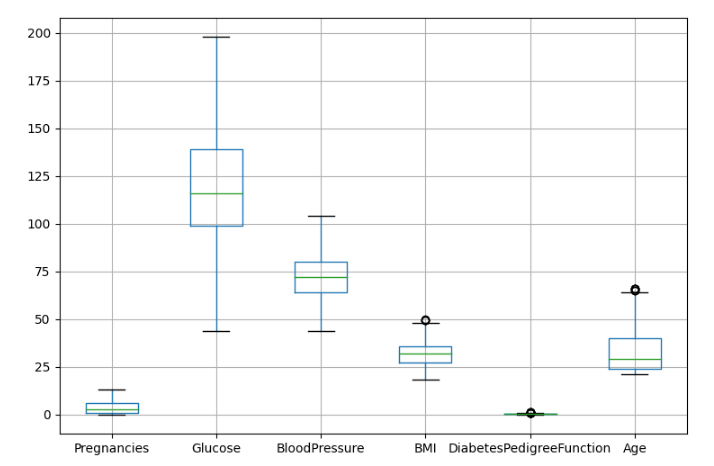
(Рис.4 Графики box-plot для параметров в которых не учитываем измененные значения)

В этих параметрах также имеются выбросы. Теперь попробуем исключить все строки таблицы, в параметрах которых есть выбросы. Так мы рискуем исключить большое количество входных данных, и если по итогу мы исключим большую часть таблицы, то потом вернемся к этому этапу и будем изменять значение выбросов на медианное значение.

Для начала создадим функцию, которая будет искать минимальное и максимальное значение границы значений, и если значения не будут входить в эту границу, то мы будем исключать их из таблицы. В функцию в качестве аргумента будет передаваться массив данных, у которого сначала найдем первый и третий квартили, после чего искать их разницу, умножать на 1.5 и отнимать это значение от первого квартиля и прибавлять ко второму, таким образом получим два значения – границы допустимых значений. На графике box-plot они обозначены усами. Заметим, что в некоторых случаях усы имеют разную длину, это происходит потому что параметр ограничен, например, в случае с инсулином, нижняя граница должна была быть отрицательна, но сам параметр ограничен значением близким к нулю снизу. Мы не будем вводить такие ограничения, так-как они никак не повлияют на наличие выбросов (если параметр имеет, например, минимальное значение до нижнего конца уса, то сам ус не продолжается, а останавливается на этом минимальном значении).

Теперь найдем границы допустимых значений для каждого из параметров кроме SkinThickness и Insulin. Для этого вызовем у таблице метод apply() и передадим функцию поиска границ, таким образом функция применится к каждой колонке и мы получим таблицу минимальных и максимальных допустимых значений.

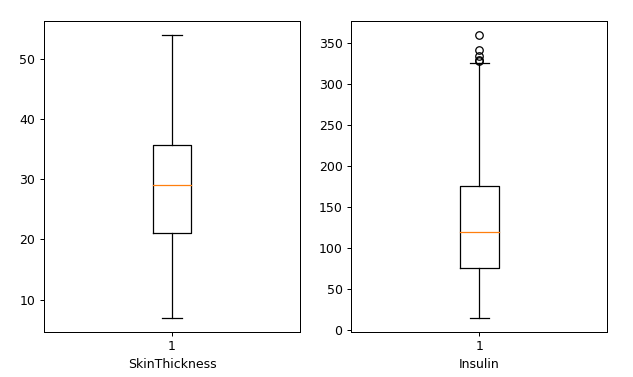
Для того, чтобы исключить выбросы воспользуемся маской размерности равной количеству строк в таблице, по итогу, например, если значение маски с индексом два будет ложью, значит строку с индексом два в таблице нужно исключить. Пройдемся в цикле по всем параметрам и на каждой итерации будем по каждому параметру создавать маску с истинным значением в том случае, если значение ячейки находится в пределах соответственного интервала. После применим логическое умножение к двум этим маскам. Таким образом в конце значение True будет лишь у тех строк в которых ни один параметр не имел выброса и просто перезапишем нашу базу данных. Построим графики box-plot для новой таблицы (рис. 5) и заметим, что действительно удалось избавиться от выбросов, присутствовавших на предыдущем графике, но при этом на границах появились новые выбросы. Так произошло потому что новый график строился без учета тех значений, которые мы исключили и соответственно значения, находившиеся вблизи границы на предыдущем графике теперь отображаются как выбросы. Но такие значения выбросами не считаются, потому что входят в интервал допустимых значений до исключения выбросов.



(Рис. 5 Графики box-plot, после того как исключили выбросы)

Важное замечание, теперь в таблице стало меньше строк и стоит обновить массивы с индексами непустых ячеек инсулина и толщины кожи, потому что некоторые из индексов, в которых значения были непустые могли быть потеряны при исключении выбросов. Для этого приведем списки с индексами к множеству set – мы можем так сделать, так как каждый элемент в массивах являлся уникальным индексом, после чего приведем оставшиеся индексы базы данных к множеству set и проведем логическое умножение между этими списками, таким списками. В результате получим два новых массива с индексами непустых значений, в которых также будут исключены те элементы, от которых мы отказались при исключении выбросов.

Теперь аналогично избавимся от выбросов для двух оставшихся параметров, при этом исключив пропущенные значения. В этот раз рассчитаем границы допустимых значений отдельно для каждого из двух параметров, так как индексы не пустых значений у параметров различны, после чего с помощью маски определим какие из строк в базе данных стоит исключить. Перезапишем базу данных и снова обновим списки с индексами непустых значений. Построим графики и посмотрим удалось ли избавится от выбросов (рис.6). По графику видно, что произошла аналогичная ситуация с изменением границ допустимых значений и появления из-за этого новых выбросов, но если обратиться к предыдущему графику, можно заметить, что интервал допустимых значений выходит за рамки значения 350 сверху, и значит те выбросы, что образовались на новом графике входят в область допустимых значений.



(Рис. 6 Графики box-plot, после того как исключили выбросы для SkinThickness и Insulin)

­

**Обучение модели**

Для начала проинициализируем все необходимые библиотеки. Обучение модели будем производить с помощью алгоритма Random Forest, при этом попробуем применить при обучении три различных тренировочных базы данных: с заменой пропущенных значений на медиану, с заменой пропущенных значений на медиану и удалением столбцов SkinThickness и Insulin, а также базу данных не включающую пропущенные значения SkinThickness и Insulin. Для каждого способа обучения будем использовать единый набор параметров, подбираемых с помощью кросс-валидации.

Основными параметрами будут являться: количество деревьев, которые будут построены, максимальная глубина деревьев, минимальное количество элементов, необходимое для разделения внутреннего узла, минимальное количество элементов, которое требуется для конечного узла и функция оценки энтропии в узлах. Увеличение количества деревьев в нашем «лесу», не вызывает переобучения модели, а лишь уменьшение вероятности получить случайный результат, так как с большим количеством деревьев их результат усредняется. Поэтому изначально установим достаточно большое число, так-как при подборе этого параметра понадобиться очень большое количество времени. Минимальное количество элементов для разделения и для конечного узла будут изменяться от двух до одиннадцати, так-как слишком большое их число ведет к недостаточному обучению модели, а переобучения постараемся избежать за счет ограничения максимальной глубины деревьев и ограничим его числом четырнадцать. Критерий будет единым – энтропия.

Сам процесс обучения будет одинаковым для каждого набора входных значений, поэтому для удобства создадим функцию, на вход которой пойдет два аргумента: первый – таблица из независимых переменных, а второй – серия результатов для каждого набора переменных (Outcome). Внутри функции будет создаваться классификатор Random Forest, после чего будем обучать модель на переданных данных и с помощью GridSearchCV найдем оптимальные параметры, после чего функция вернет наилучший обученный классификатор.

Метрики во всех трех случаях также будут рассчитываться идентично, поэтому создадим функцию, для расчета значений precision, recall и f1. Заранее проинициализируем необходимые библиотеки. На вход функции будет подаваться массив с предсказаниями модели и массив с истинными значениями. После чего функция рассчитает и напечатает precision, recall и f1 для данных параметров.

Сначала попробуем обучить модель на таблице с заменой пропущенных значений. Разделим нашу таблицу на 3 части: данные на которых модель будет обучаться, валидационные данные и тестовые данные. Пусть тестовые данные будут состоять из 25% базы данных, остальные определим, как тренировочные, после чего пусть 25% тренировочных данных будут являться валидационными. Так как такое разбиение сложнее, чем то, что может дать модуль skleaern train\_test\_split, то разобьем данные самостоятельно. Для этого инициализируем массив с индексами всей оставшейся таблицы, после чего перемешаем его с помощью функции np.random.shuffle, и первые 75% данных запишем в массив train\_indexes, а остальные в test\_indexes. Аналогично разобьем train\_indexes на train\_indexes и validation\_indexes. Можно проверить не потеряли ли мы в процессе разбиения индексы, для этого просуммируем длинны всех трех массивов и получим, что их суммарная длина равна длине первоначального массива. Стоит заметить что количество строк в таблице после удаления выбросов стало равно 684, то есть избавляясь от выбросов мы всего лишь незначительно уменьшили объемы данных для обучения.

Запишем в таблицу train\_X все параметры из таблицы с заменой пустых значений, кроме результата Outcome с индексами - train\_indexes, которые получили на предыдущем шаге, а в таблицу (серию) train\_y только результаты с соответственными индексами. Аналогично инициализируем валидационные данные, при этом тестовые данные пока не имеет смысла инициализировать, так-как сравнивать результаты различных моделей будем на основе только валидационных, для того, чтобы избежать переобучения на тестовых данных. Но, стоит заменить, что для каждой модели тренировочные и тестовые индексы будут инициализироваться случайным образом, поэтому список с тестовыми индексами стоит сделать уникальным и не перезаписывать его, для того, чтобы если текущая модель покажет себя лучше остальных на валидационных данных, в последствии мы могли инициализировать её тестовые данные. Для этой модели тренировочные индексы запишем в список test\_indexes\_wrc, где wrc – without removing columns. Аналогично и классификатор не следует перезаписывать в последствии поэтому назовем его best\_clf\_wrc. Обучим модель с помощью описанной ранее функции и найдем её прогноз. Посчитаем метрики на основе этого прогноза и получим следующий результат

precision = 0.703,

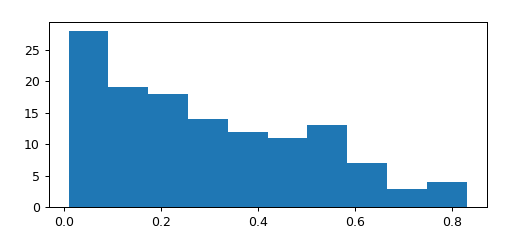
recall = 0.452,

f1 = 0.55

Метрика precision достаточно высока, но с учетом специфики задачи будем оптимизировать recall.

В задаче необходимо определить, болеет ли пациент диабетом, поэтому если стоит выбор между точностью при оптимизации которой уменьшается количество «промахов» (неверно предугаданных как истина результатов) и точностью при оптимизации которой уменьшается количество предугаданных как ложные, но при этом истинных результатов, стоит выбирать в пользу оптимизации второй метрики. Таким образом мы уменьшим количество пациентов, у которых на самом деле был диабет, но мы определили это неверно. Так мы рискуем увеличить количество пациентов, у которых диабета не было, но при этом мы сказали, что он есть, но это менее опасно для самих пациентов, так как если мы не нашли диабет, то здоровье пациента будет в опасности, но если мы нашли диабет у пациента, у которого его на самом деле нет, то такой пациент скорее всего будет отправлен на дополнительное обследование и при этом жизни и здоровью человека ничего не угрожает. Но при этом оптимизируя метрику recall, нам необходимо следить за тем, чтобы метрика f1 не была слишком низкой, потому что таким образом мы может просто диагностировать диабет вообще у всех пациентов, что было бы крайне неэффективно.

Для оптимизации метрики recall получим сначала массив из первичных предсказаний вероятности модели принадлежности примеров к тому или иному классу. Для этого у классификатора вызовем метод predict\_proba в который передадим валидационную таблицу не зависивых переменных val\_X. Отберем только второе значение, которое говорит о вероятности отнесения пациента к классу больных, в нашем случае к классу 1 и построим гистограмму частот (рис 7).



­­­(рис 7. Гистограмма частот вероятностей для модели WRC)

Заметим, что при увеличении вероятности количество элементов имеет тенденцию к убыванию. Попробуем подобрать такой порог вероятности, после которого пациенты будет считаться больны, что метрики recall и f1 будут достаточно большими. Не будем искать порог, при котором метрики будут максимально высокими, для того, чтобы потом можно было сравнить результат других моделей на валидационных данных с таким же пороговым значением Пусть например, пусть таким барьерным значение будет 0.3. Переопределим предсказание модели с учетом такого порога и снова посчитаем метрики.

precision = 0.603,

recall = 0.833,

f1 = 0.7

Удалось достаточно хорошо оптимизировать метрику recall и при этом метрика f1 тоже возросла. Остановимся на таком результате для модели WRC.

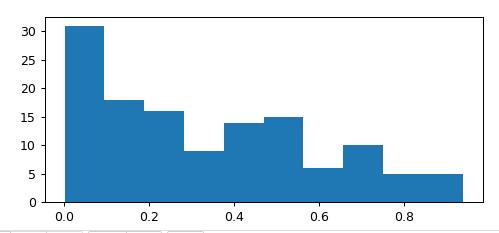
Теперь построим модель, в которой не будем учитывать столбцы SkinThickness и Insulin. Аналогично предыдущей модели создаем массив с индексами тренировочной таблицы, перемешиваем и по аналогичному алгоритму разделяем данные на тренировочные, валидационные и тестовые. Также сохраняем индексы тестовых данных в переменной, которую потом не будем переопределять и назовем её test\_indexes\_rc (removing columns). Создаем массивы тренировочных и валидационных данных, но в этот раз, с помощью метода drop в таблицах с независимыми переменными train\_X и val\_X, убираем не только параметр Outcome, но и SkinThickness и Insulin. Тренируем модель на тестовых данных и получаем классификатор с оптимальными параметрами. Для начала посчитаем все метрики для предсказания данного классификатора на валидационных данных и получаем результат:

precision = 0.667

recall = 0.571

f1 = 0.615

Сразу получили достаточно хорошие метрики, но все равно постараемся оптимизировать recall, так как в задаче важна именно эта метрика. Возьмем предсказание вероятностей отношения пациента к классу 1 и построим гистограмму частот (рис.8)



(рис 8. Гистограмма частот вероятностей для модели RC)

Получили гистограмму схожую с аналогичной гистограммой предыдущей модели, только в этот раз график стал менее гладким и появились провалы в значениях примерно 0.35 и 0.6. Возможно они связаны с тем, что в выборке стало меньше параметров. И учитывая, что графики распределения вероятностей похожи, можем сравнить метрики моделей при равных пороговых значениях вероятности, при которой пациент будет считаться больным. Тогда аналогичным образом переопределим прогноз модели с пороговым значением 0.3 и получим следующие результаты.

precision = 0.556

recall = 0.833

f1 = 0.667

В результате получили recall равный recall предыдущей модели, но при этом метрика precision и соответственно f1 выросли. Значит можно сделать вывод, что модель WRC на валидационных данных показала себя лучше, чем RC.

Попробуем создать модель, которая не будет включать пропущенные значения в параметрах SkinThickness и Insulin. Для этого, приводим массивы индексов непустых значений этих параметров к структуре set, после чего производим логическое умножение, чтобы найти общие индексы, такие, что в каждой строке не будет пропущенных ячеек ни в одном из параметров. После преобразовываем получившийся сет в массив numpy и, как и предыдущих моделях перемешиваем его и выделяем тестовые, тренировочные и валидационные данные. Индексы тестовых данных назовем test\_indexes\_wev, где wev – without empty values. Проверим на сколько уменьшилась тренировочная таблица. В таблице осталось 339 строк, 25% из которых будут являться тестовыми, и ещё 25% из оставшихся будут валидационными. После всех преобразований мы сильно уменьшили объемы данных. Главная проблема такой модели в том, она требует, чтобы у пациента были известны все параметры. Значит в случае, если параметры SkinThickness и Insulin в тестовых данных будут неизвестны, модель не сможет произвести прогноз.

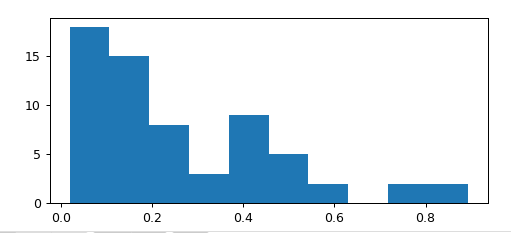
Обучим модель с помощью функции training\_model на тестовых данных, после найдем прогноз и посчитаем метрики.

precision = 0.75

recall = 0.6

f1 = 0.667

Как и в двух других моделях получаем что precision больше чем recall и f1, для оптимизации recall для начала построим гистограмму частот вероятностей отнесения пациента к классу 1.



(рис 9. Гистограмма частот вероятностей для модели WEV)

Получили график похожий на аналогичный график другим моделям, только, видимо из-за уменьшения количества обучающих данных на графике появились провалы, и гистограмма стала ещё менее в сравнении с гистограммой модели RC. Но учитывая общую тенденцию и сходство графиков можем аналогичным образом найти метрики с таким же барьером и сравнить результаты:

precision = 0.409

recall = 0.9

f1 = 0.563

С таким барьером, получили достаточно высокое значение recall, выше чем в предыдущих моделях, но при этом значение метрики f1 стало значительно ниже. Попробуем скорректировать барьер для этой модели, чтобы сделать одну из метрик примерно равной тем, что были в предыдущих моделях, чтобы можно был наглядно их сравнить.

При значении барьера 0.4, получаем следующие метрики:

precision = 0.5625

recall = 0.9

f1 = 0.692

Но из-за малого количества данных такой результат может быть случайным и подогнанным специально под валидационные данные. Также учитывая, что если в новых тестовых данных будут пропущенные значения, по аналогии с первоначальной таблицей, то такая модель не сможет сделать прогноз при наличии у пациента пропущенного значения. Поэтому в результате лучшей моделью WRC, так как она универсальна и дает хорошие метрики.

**Проверка модели на тестовых данных**

Для проверки модели на тестовых данных для начала инициализируем тестовые данные, с помощью сохраненных списков индексов тестовых данных для модели WRC. Так как в обучении модели участвуют все параметры первоначальной таблицы, в таблицу test\_X запишем все колонки кроме Outcome, а в серию test\_y только Outcome.

Теперь дня начала посмотрим точность модели, то есть у какого процента пациентов она верно определяет их класс. Для этого сначала найдем вероятности определения пациентов к классу больных. После переопределим прогноз модели с учетом барьера вероятности равного 0.3, аналогично тому, как мы переопределяли его для валидационных данных. Важно, что барьер является гиперпараметром, который на этом этапе нельзя изменять, чтобы избежать переобучения модели конкретно под такую тестовую выборку. Теперь воспользуемся функцией accuracy\_score из библиотеки sklearn, в которую передадим прогноз модели и правильный результат – серию test\_y. Получаем хорошую точность равную 0.784. Теперь найдем остальные метрики и получим следующий результат:

precision = 0.667

recall = 0.806

f1 = 0.73

**Вывод**

С помощью модели, которая при обучении заменяет пропущенные значения в таблице на медианные при этом обучается с помощью алгоритма Random Forest удалось получили достаточно высокие значение recall и f1 на тестовых данных, заметим, что результат модели на валидационных данных схож с тем, что мы получили на тестовых данных, это значит, что модель не переобучена, и при этом дает достаточно точный и стабильный результат на новых данных.