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 __Информатика, искусственный интеллект и системы управления_____

КАФЕДРА ____Системы обработки информации и управления_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

_Прогнозирование инсульта_____

Студент ___ИУ5-62Б____ _____ __**Васильченко Д.Д.**__
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ ____**Гапанюк Ю.Е.**___
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2022 г.

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)

« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме ____Прогнозирование инсульта_____

_____

Студент группы ___ИУ5-62Б___

____Васильченко Дарья Дмитриевна_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_____

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения НИР: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Техническое задание *__Решить задачу машинного обучения. Построить модели машинного обучения для решения задачи классификации. Должно быть выбрано не менее пяти моделей, две из которых должны быть ансамблевыми. Построить базовое решение для выбранных моделей без подбора гиперпараметров и решение с найденными оптимальными значениями гиперпараметров. Сравнить качество полученных моделей с качеством baseline-моделей. Оценить качество построенных моделей на основе выбранных метрик. Должно быть выбрано не менее трех метрик*_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _27_ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель НИР _____ **___Гапанюк Ю.Е.____**
(Подпись, дата) (И.О.Фамилия)

Студент _____ **__Васильченко Д.Д.__**
(Подпись, дата) (И.О.Фамилия)

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

Содержание

1. Введение

Машинное обучение - обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться. ML обучает компьютерную систему составлению точных прогнозов при вводе данных. Машинное обучение – одна из наиболее распространенных форм применения искусственного интеллекта современным бизнесом.

В основе машинного обучения лежат три компонента:

- Данные. Собираются всевозможными способами. Чем больше данных, тем эффективнее машинное обучение.
- Признаки. Определяют, на каких параметрах строится машинное обучение.
- Алгоритм. Выбор метода машинного обучения будет влиять на точность, скорость работы и размер готовой модели.

К основным задачам машинного обучения относятся:

- Регрессия. Предоставляет прогноз на основе выборки объектов с различными признаками. По итогам анализа данных на выходе получается число или числовой вектор.
- Классификация. Выявляет категории объектов на основе имеющихся параметров.
- Кластеризация. Разделяет данные на схожие категории по объединяющему признаку.
- Идентификация. Отделяет данные с заданными параметрами от остального массива данных.
- Прогнозирование. Работает с объемами данных за определенный период и предсказывает на основе анализа их значение через заданный период времени.
- Извлечение знаний. Исследует зависимости между рядом показателей одного и того же явления или события.

Машинное обучение уже применяется во всех сферах деятельности человека. Например, в робототехнике, маркетинге, обеспечении безопасности, финансовом секторе, добыче полезных ископаемых и медицине.

Данная работа посвящена решению задачи машинного обучения в медицинской сфере - задачи прогнозирования инсульта.

2. Описание набора данных

Был выбран набор данных о пациентах «Stroke Prediction Dataset» со следующими признаками:

1) id: уникальный идентификатор пациента
2) gender: пол пациента (“Male”, “Female”, “Other”)
3) age: возраст пациента
4) hypertension: наличие гипертонии у пациента (0, если у пациента нет гипертонии, 1, если у пациента есть гипертония)
5) heart_disease: наличие сердечных заболеваний у пациента (0, если у пациента нет сердечных заболеваний, 1, если у пациента есть сердечные заболевания)
6) even_married: состоит ли пациент в браке (“No”, “Yes”)
7) work_type: работает ли пациент, если работает, то как (“children”, “Govt_jov”, “Never_worked”, “Private”, “Self-employed”)
8) Residence_type: место проживания пациента (“Rural”, “Urban”)
9) avg_glucose_level: средний уровень глюкозы в крови пациента
10) bmi: индекс массы тела пациента
11) smoking_status: отношение пациента к курению (“formerly smoked”, “never smoked”, “smokes”, “Unknown”)
12) stroke – целевой признак датасета: был ли у пациента инсульт (1, если у пациента был инсульт, 0 , если у пациента не было инсульта)

3. Постановка задачи

В данной работе будет решаться задача бинарной классификации на примере прогнозирования вероятности инсульта у пациента.

Задача классификации — задача, в которой имеется множество объектов (ситуаций), разделённых, некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества. Классифицировать объект — значит, указать номер (или наименование) класса, к которому относится данный объект. Классификация объекта — номер или

наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

Таким образом, задача заключается в прогнозировании параметра stroke. Её решение включает в себя следующие этапы:

1) Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
2) Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
3) Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения.
4) Выбор метрик для последующей оценки качества моделей.
5) Выбор наиболее подходящих моделей для решения задачи классификации.
6) Формирование обучающей и тестовой выборок на основе исходного набора данных.
7) Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров.
8) Подбор гиперпараметров для выбранных моделей.
9) Построение решения для найденных оптимальных значений гиперпараметров.

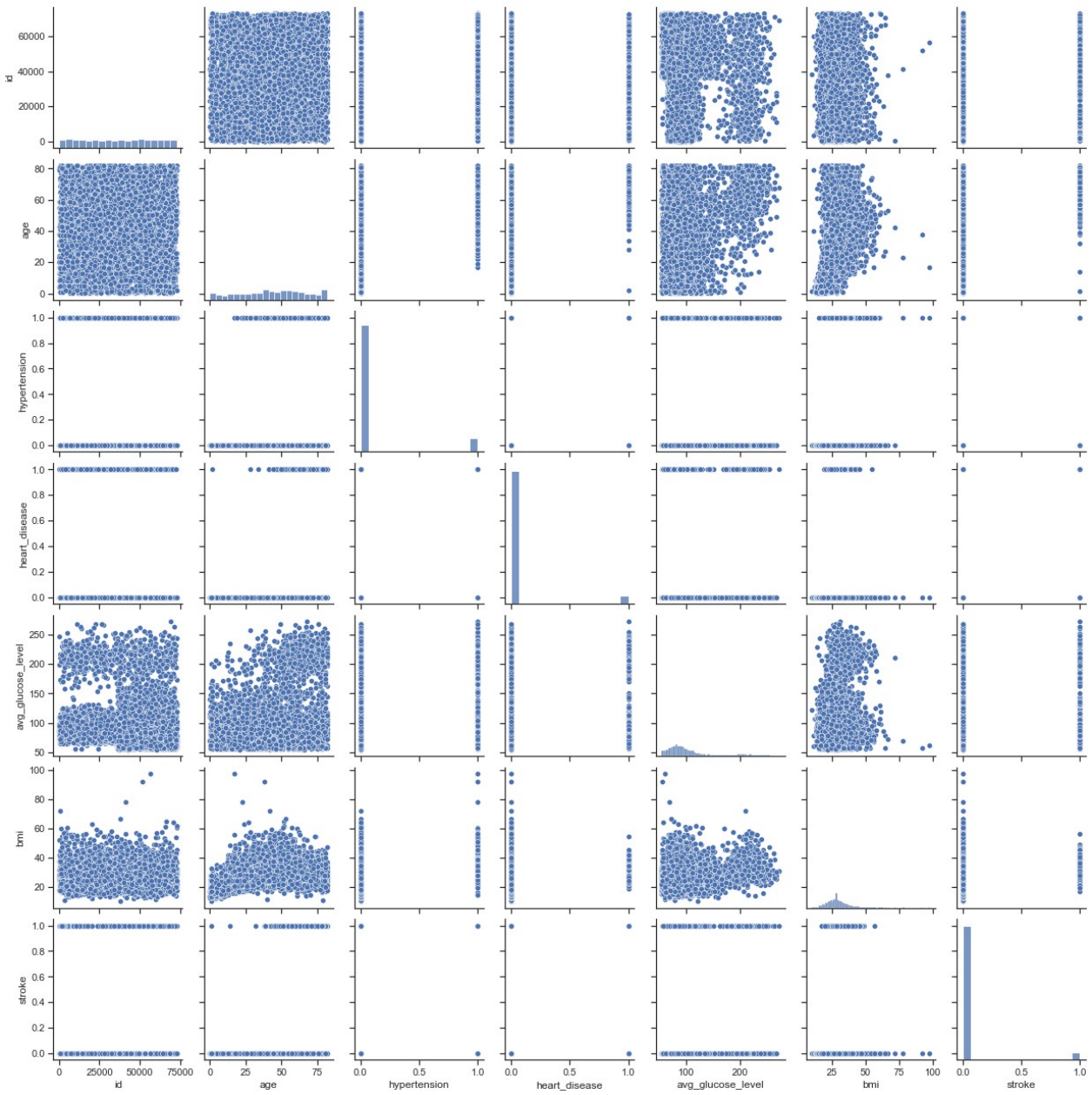
4. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных.

Разведочный анализ данных (Exploratory Data Analysis) – предварительное исследование датасета с целью определения его основных характеристик, взаимосвязей между признаками, а также сужения набора методов, используемых для создания модели машинного обучения.

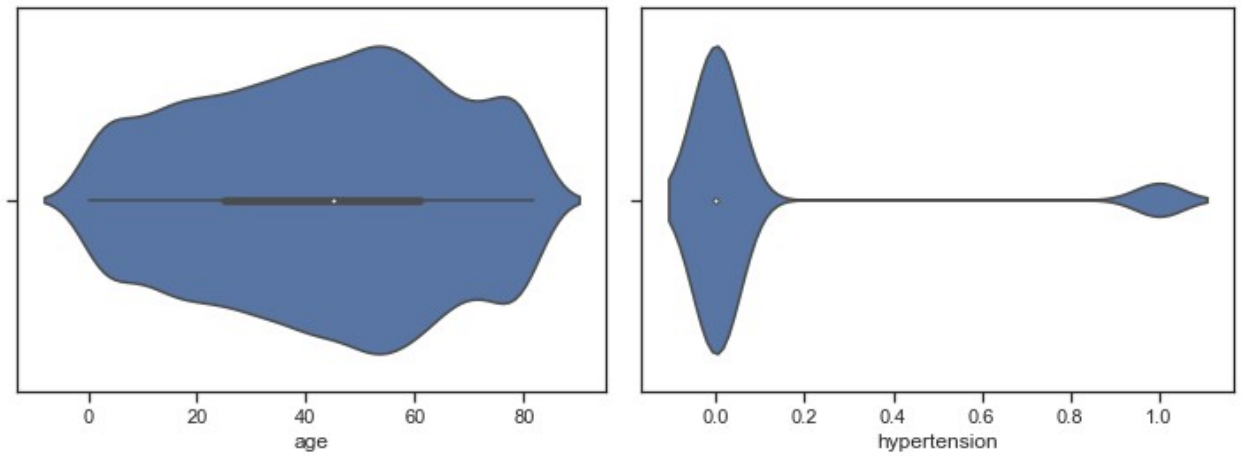
При проведении разведочного анализа данных на выбранном датасете были выявлены следующие его особенности.

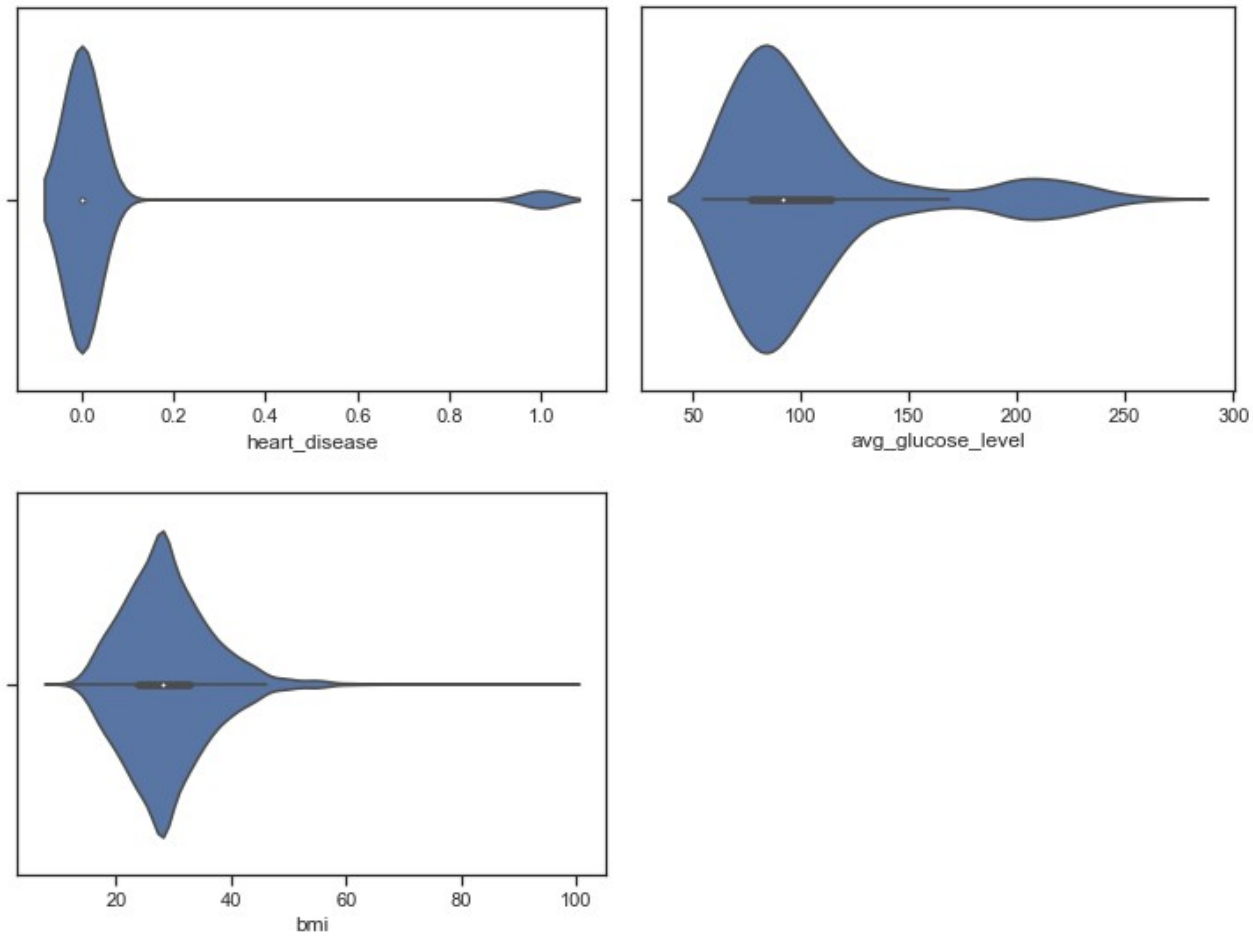
1) После удаления дубликатов записей, размер датасета - 5110 строк, 12 колонок.
2) В датасете представлены данные следующих типов: int64, object, float64.
3) В датасете есть признак id, содержащий только уникальные значения, его при построении моделей машинного обучения использовать не будем, так как никакой зависимости целевого признака от полностью уникального признака быть не может.
4) Значение «Other» признака gender составляет очень незначительную долю от всего количества записей (1 из 5110). Удалим единственную строку с таким значением, т.к. оно не пригодится для дальнейшей работы.
5) Датасет содержит пропуски в значениях признака bmi, их необходимо устранить.
6) Целевой признак stroke для задачи бинарной классификации, действительно, содержит только 0 и 1.
7) Присутствует довольно сильный дисбаланс классов целевого признака stroke. Класс 0 составляет 95,13%, а класс 1 составляет 4,87%. Такой дисбаланс следует сбалансировать.

Также были построены парные диаграммы для понимания структуры данных:



Были построены скрипичные диаграммы для числовых колонок (age, hypertension, heart_disease, avg_glucose_level, bmi):





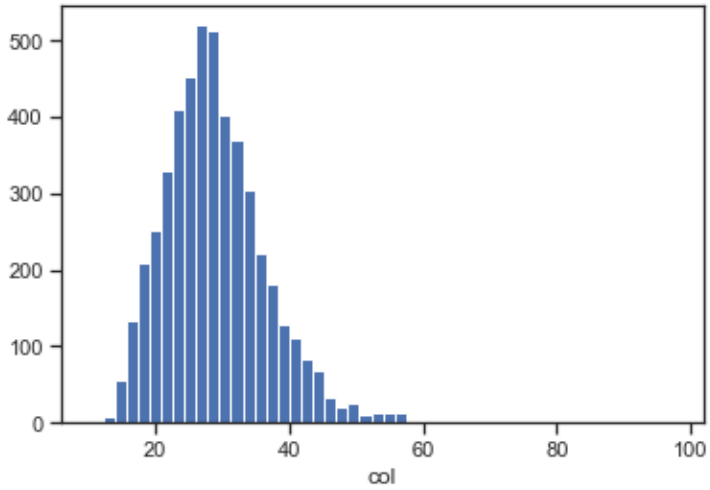
5. Анализ и заполнение пропусков в данных.

Один из простых способов решения проблемы пропусков в данных - просто игнорировать или удалять строки, в которых отсутствуют данные, выбрасывая их из нашего анализа. Однако этот метод может быть плох из-за потери информации.

Еще один способ — это заполнение пропусков, где отсутствующее значение заменяется каким-либо образом. Базовые реализации заменят все отсутствующие значения средним, медианным, либо же константой.

В данной работе были использованы встроенные средства импьютации библиотеки scikit-learn. С помощью класса SimpleImputer была проведена импьютация средним значением.

Гистограмма по признаку bmi:



Итак, представленный набор данных содержал пропуски в значениях признака `bmi`, которые были обработаны путем их заполнения средним значением.

6. Кодирование категориальных признаков.

Большинство алгоритмов машинного обучения не могут обрабатывать категориальные переменные, если они не преобразованы в числовые значения, а производительность многих алгоритмов зависит от того, как закодированы категориальные переменные.

В случае кодирования категорий целочисленными значениями уникальные значения категориального признака кодируются целыми числами.В scikit-learn для такого кодирования используется два класса: LabelEncoder и OrdinalEncoder.

LabelEncoder ориентирован на применение к одному признаку. Этот класс прежде всего предназначен для кодирования целевого признака, но может быть также использован для последовательного кодирования отдельных нецелевых признаков.

OrdinalEncoder ориентирован на применение к матрице объект-признак, то есть для кодирования матрицы нецелевых признаков.

LabelEncoder и OrdinalEncoder могут использоваться только для категориальных признаков в номинальных шкалах (для которых отсутствует порядок), например города, страны, названия рек и т.д. Это связано с тем, что задать какой-либо порядок при кодировании с помощью LabelEncoder и OrdinalEncoder невозможно, они сортируют категории в лексикографическом порядке.

Так как в нашем случае, для категориальных признаков отсутствует порядок, для кодирования категориальных признаков был использован класс LabelEncoder.

В представленном датасете присутствуют следующие категориальные признаки: gender, ever_married, work_type, Residence_type, smoking_status. Данные признаки содержат следующие классы:

1) Уникальные значения признака gender: ['Male' 'Female']
2) Уникальные значения признака ever_married: ['Yes' 'No']
3) Уникальные значения признака work_type: ['Private' 'Self-employed' 'Govt_job' 'children' 'Never_worked']
4) Уникальные значения признака Residence_type: ['Urban' 'Rural']
5) Уникальные значения признака smoking_status: ['formerly smoked' 'never smoked' 'smokes' 'Unknown']

После кодирования категориальных признаков в датасете остались данные следующих типов: int64, float64.

7. Масштабирование данных.

Масштабирование предполагает изменение диапазона измерения величины.

Если признаки лежат в различных диапазонах, то необходимо их нормализовать. Как правило, применяют два подхода:

- MinMax масштабирование:

$$x_{\text{новый}} = \frac{x_{\text{старый}} - min(X)}{max(X) - min(X)}$$

В этом случае значения лежат в диапазоне от 0 до 1.

- Масштабирование данных на основе Z-оценки:

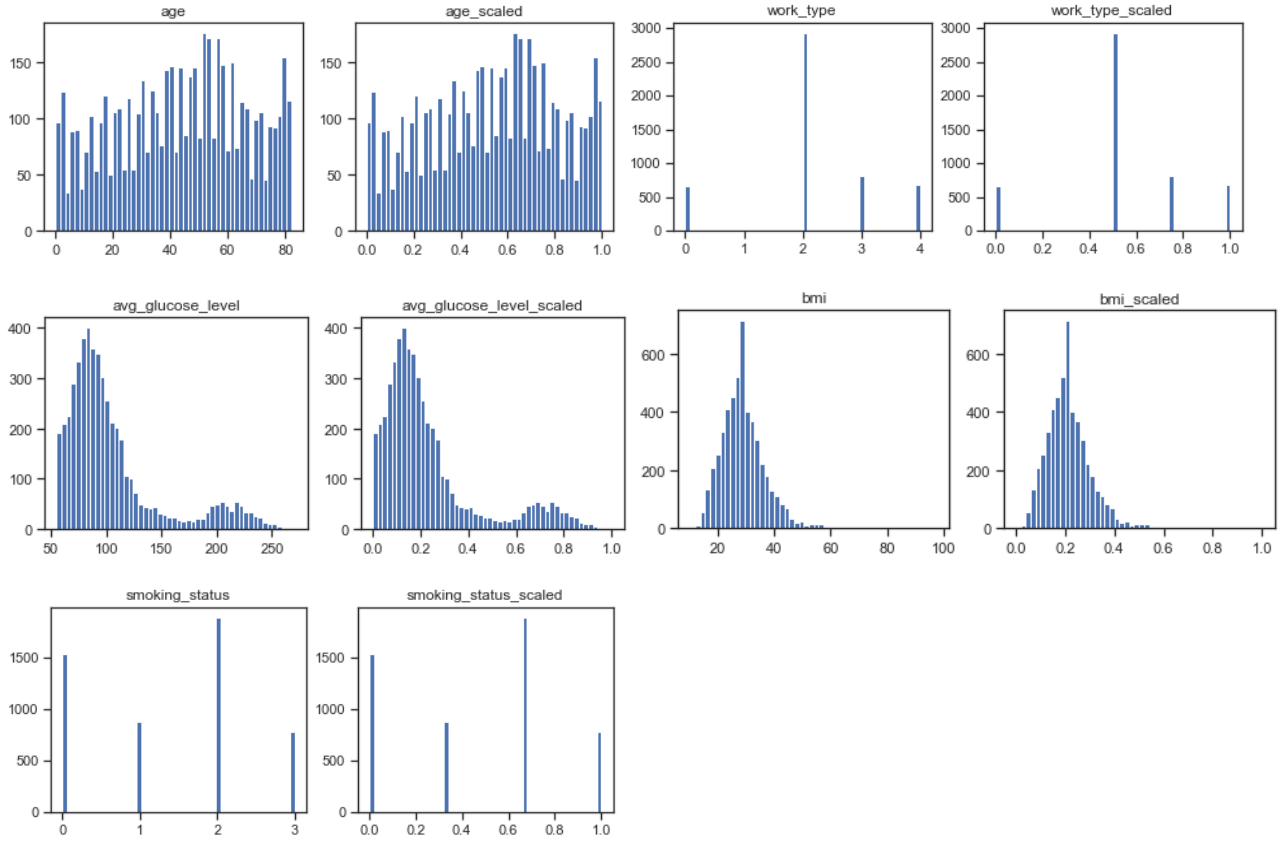
$$x_{\text{новый}} = \frac{x_{\text{старый}} - AVG(X)}{\sigma(X)}$$

В этом случае большинство значений попадает в диапазон от -3 до 3.

где X - матрица объект-признак, AVG(X) - среднее значение, σ - среднеквадратичное отклонение.

В данной работе использован метод MinMaxScaler из библиотеки sklearn. Были масштабированы следующие колонки: age, work_type, avg_glucose_level, bmi, smoking_status.

Убедились, что масштабирование не повлияло на распределение данных:



8. Проведение корреляционного анализа данных. Выбор признаков, подходящих для построения моделей.

Корреляционная зависимость – это согласованные изменения двух (парная корреляционная связь) или большего количества признаков (множественная корреляционная связь). Суть ее заключается в том, что при изменении значения одной переменной происходит закономерное изменение (уменьшение или увеличение) другой(-их) переменной(-ых).

Корреляционный анализ – статистический метод, позволяющий с использованием коэффициентов корреляции определить, существует ли зависимость между переменными и насколько она сильна.

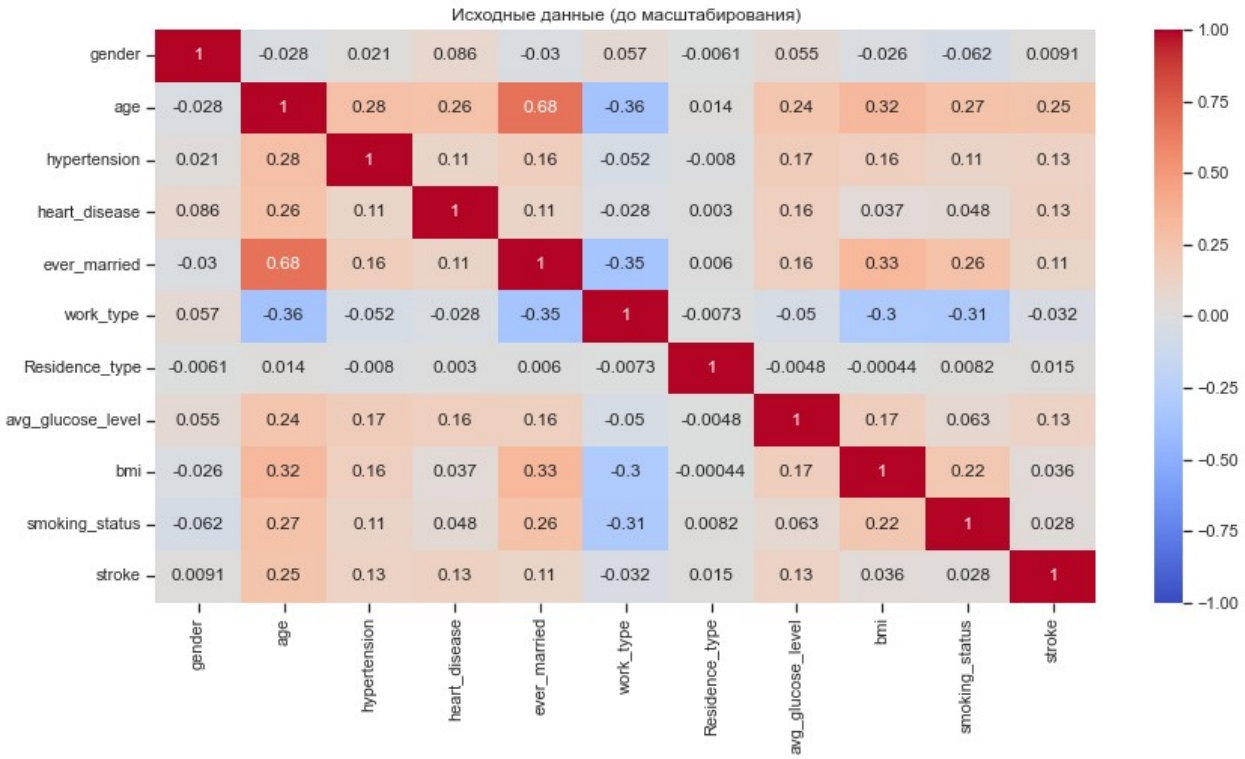
Коэффициент корреляции – двумерная описательная статистика, количественная мера взаимосвязи (совместной изменчивости) двух переменных.

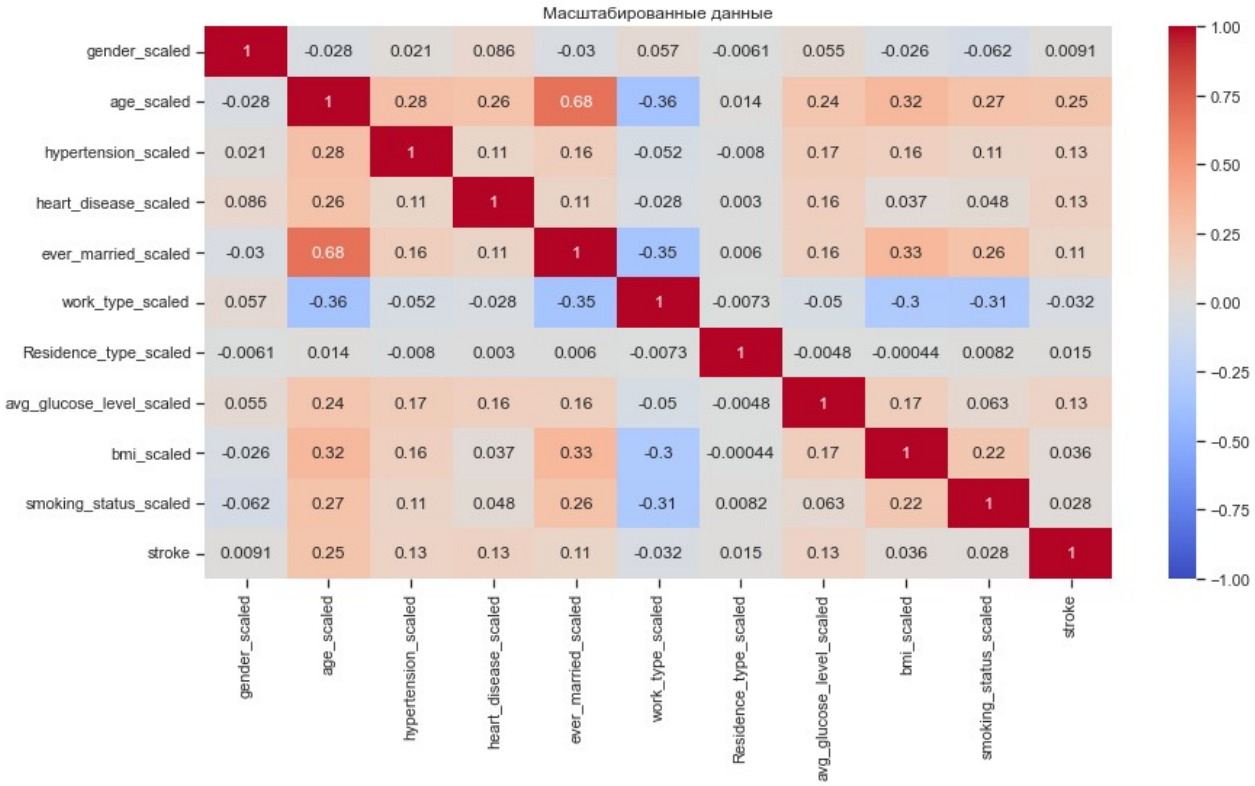
Корреляционный анализ применяется для количественной оценки взаимосвязи двух наборов данных, представленных в безразмерном виде. Корреляционный анализ дает возможность установить, ассоциированы ли наборы данных по величине. Коэффициент корреляции, всегда обозначаемый латинской буквой r, используется для определения наличия взаимосвязи между двумя свойствами.

При положительной линейной корреляции более высоким значениям одного признака соответствуют более высокие значения другого, а более низким значениям одного признака – низкие значения другого.

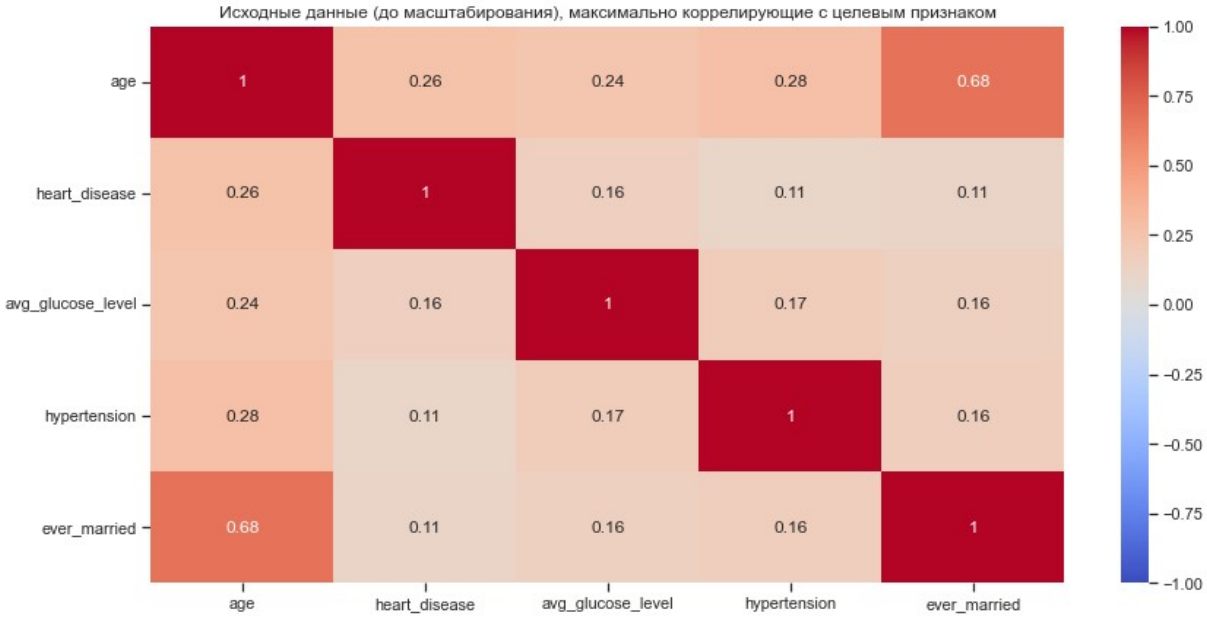
При отрицательной линейной корреляции более высоким значениям одного признака соответствуют более низкие значения другого, а более низким значениям одного признака – высокие значения другого.

В выбранном датасете провели корреляционный анализ по исходным данным (до масштабирования) и по масштабированным данным:





Определили признаки, имеющие максимальную по модулю корреляцию с целевым признаком (>0.1): age, heart_disease, avg_glucose_level, hypertension, ever_married. Построили корреляционную матрицу для этих признаков:



На основе корреляционной матрицы можно сделать следующие выводы:

- Корреляционные матрицы для исходных и масштабированных данных совпадают.
- Целевой признак классификации stroke наиболее сильно коррелирует с возрастом пациента (0.25), наличием или отсутствием сердечных заболеваний у пациента (0.14), средним уровнем глюкозы в крови

пациента (0.13), наличием или отсутствием гипертонии у пациента (0.13), состоял ли пациент в браке (0.11). Эти признаки обязательно следует оставить в модели классификации.

- Небольшие по модулю значения коэффициентов корреляции свидетельствуют о незначительной корреляции между исходными признаками и целевым признаком.

9. Выбор метрик.

В качестве метрик для решения задачи классификации будем использовать:

1) Метрика precision:

$$precision = \frac{TP}{TP+FP}$$

Доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

Используется функция precision_score.

2) Метрика recall (полнота):

$$recall = \frac{TP}{TP+FN}$$

Доля верно предсказанных классификатором положительных объектов, из всех действительно положительных объектов.

Используется функция recall_score.

3) Метрика ***F*1**-мера

Для того, чтобы объединить precision и recall в единую метрику используется $F\beta$-мера, которая вычисляется как среднее гармоническое от precision и recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{precision + recall}$$

где β определяет вес точности в метрике.

На практике чаще всего используют вариант F1-меры (которую часто называют F-мерой) при β=1:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Для вычисления используется функция f1_score.

4) Метрика ROC AUC

Основана на вычислении следующих характеристик:

$TPR = \frac{TP}{TP+FN}$ - True Positive Rate, откладывается по оси ординат. Совпадает с recall.

$FPR = \frac{FP}{FP+TN}$ - False Positive Rate, откладывается по оси абсцисс. Показывает, какую долю из объектов отрицательного класса алгоритм предсказал неверно.

Идеальная ROC-кривая проходит через точки (0,0)-(0,1)-(1,1), то есть через верхний левый угол графика.

Чем сильнее отклоняется кривая от верхнего левого угла графика, тем хуже качество классификации.

В качестве количественной метрики используется площадь под кривой - ROC AUC (Area Under the Receiver Operating Characteristic Curve). Чем ниже проходит кривая, тем меньше ее площадь и тем хуже качество классификатора.

Для получения ROC AUC используется функция roc_auc_score.

10. Выбор моделей машинного обучения.

Для задачи классификации будем использовать следующие модели:

1) Логистическая регрессия

Метод основан на использовании логистической кривой или сигмоиды:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Модель линейной регрессии

$$z(x) = f(x, b) = b_0 + b_1 \cdot x_1 + \cdots + b_n \cdot x_n$$

(диапазон z(x)$\in(-\infty,\infty)$)

может быть использована как параметр сигмоиды:

$$h(z) = \frac{1}{1 + e^{-z}}$$

(диапазон h(z)$\in$(0,1))

Таким образом, h(z) можно рассматривать как вероятность принадлежности объекта x_i к единичному классу. Вероятность принадлежности объекта x_i к нулевому классу $1-h(z)$.

Подставляя параметр z(x) в функцию h(x)=h(z(x)), получаем:

$$h(x) = \frac{1}{1 + e^{-(b_0 + b_1 \cdot x_1 + \cdots + b_n \cdot x_n)}}$$

Используем следующую функцию потерь:

$$cost = \begin{cases} -ln(h(x)), y = 1 \\ -ln(1 - h(x)), y = 0 \end{cases}$$

Такой вид функции используется для того, чтобы создать максимально сильный штраф при несопадении истинного и предсказанного классов.

Комбинируя оба условия, получаем:

$$cost(h(x), y) = -y \cdot ln(h(x)) - (1 - y) \cdot ln(1 - h(x))$$

Формируем полную функцию потерь (параметры b неявно входят в h(xi)):

$$L(b) = -\frac{1}{k} \cdot \sum_{i=1}^{k} \Big(y_i \cdot ln(h(x_i)) + (1 - y_i) \cdot ln(1 - h(x_i)) \Big) =$$

$$L(b) = -\frac{1}{k} \cdot \sum_{i=1}^{k} \left(y_i \cdot ln\left(\frac{1}{1 + e^{-(b_0 + b_1 \cdot x_1 + \cdots + b_n \cdot x_n)}} \right) + (1 - y_i) \cdot ln\left(1 - \frac{1}{1 + e^{-(b_0 + b_1 \cdot x_1 + \cdots + b_n \cdot x_n)}} \right) \right)$$

Частные производные полной функции потерь по коэффициентам регресии b^j:

$$\frac{\partial L}{\partial b^j} = \frac{1}{k} \cdot \sum_{i=1}^{k} (h(x_i) - y_i) \cdot x_i^j$$

Если набор данных состоит из двух признаков, то он может быть визуализирован на плоскости.

В этом случае граничное условие:

$$b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 = 0$$

Тогда:

$$x_2 = -\frac{b_0 + b_1 \cdot x_1}{b_2}$$

2) Машина опорных векторов

Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

Вектор - точка в многомерном пространстве. Соответствует одному объекту обучающей или тестовой выборки. В рассматриваемом примере это двухмерное пространство (x1,x2)(x1,x2)

Опорный вектор - такая точка, которая участвует в построении оптимальной гиперплоскости.

Гиперплоскость - подпространство с размерностью, на единицу меньшей, чем объемлющее пространство. Для двухмерного пространства (плоскости) частным случаем гиперплоскости является прямая, для трехмерного пространства гиперплоскостью является плоскость.

Разделяющая гиперплоскость - гиперплоскость, которая каким-то образом разделяет вектора (точки в многомерном пространстве), принадлежащие различным классам. Возможно, разделяет с ошибками.

Оптимальная (разделяющая) гиперплоскость - такая разделяющая гиперплоскость, которая обеспечивает максимальный зазор (margin) между векторами (точками) различных классов.

3) Решающее дерево

Алгоритм построения обучающего дерева сводится к нескольким пунктам:

- Для текущего выбранного признака (колонки) из N признаков построить все варианты ветвления (разбиения) по значениям (для категориальных признаков) или по диапазонам значений (для числовых признаков). При этом будет сформировано K поддеревьев (где K - число ветвлений).

Каждое поддерево содержит подвыборку, которая включает только строки выборки, соответствующие результатам ветвления. В каждом поддереве расположена:

- или выборка, содержащая N-1 признак, если признак, для которого строится ветвление, полностью пропадает в результате ветвления;
- или выборка, содержащая N признаков, если признак, для которого строится ветвление, не пропадает полностью в результате ветвления, но при этом число строк в выборке уменьшается;

- Если подвыборке соответствует единственное значение целевого признака, то в дерево добавляется терминальный лист, который соответствует предсказанному значению.
- Если в подвыборке больше одного значения целевого признака, то предыдущие пункты выполняются рекурсивно для подвыборки.

В экспертной системе правила формируются экспертом в предметной области или на основании известных эксперту теоретических положений (дедукция) или на основании его личного исчерпывающего опыта принятия решений (полная индукция.) Дерево решений строит правила на основе обучающей выборки, которая заведомо не может содержать все знания о предметной области (неполная индукция).

Преимущества деревьев решений:

- Работают по принципу "белого ящика". Логика построенного дерева хорошо отображается на исследуемую предметную область.
- Можно визуализировать алгоритм в виде дерева.
- Требуют мало данных для обучения.
- Работает с числовыми и категориальными признаками. В настоящее время scikit-learn требует кодирования категориальных признаков с использованием LabelEncoder.

Недостатки деревьев решений:

- Могут переобучаться. Для борьбы с переобучением используется регулирование глубины дерева или "стрижка" дерева. Англ. decision_tree_pruning.

- Очень сильно зависят от набора данных в обучающей выборке. Появление одного нового примера может полностью перестроить весь каскад условий. Однако, на этом недостатке построено использование дерева в ансамблевых классификаторах, поэтому в ансамблях на основе деревьев данный недостаток можно рассматривать как достоинство.

4) Случайный лес

Случайный лес можно рассматривать как алгоритмом бэггинга над решающими деревьями.

Но при этом каждое решающее дерево строится на случайно выбранном подмножестве признаков. Эта особенность называется "feature bagging" и основана на методе случайных подпространств.

Метод случайных подпространств позволяет снизить коррелированность между деревьями и избежать переобучения. Базовые алгоритмы обучаются на случайно выбранных подмножествах признаков. Ансамбль моделей, использующих метод случайного подпространства, можно построить, используя следующий алгоритм:

1. Пусть количество объектов для обучения равно NN, а количество признаков DD.
2. Выбирается число отдельных моделей в ансамбле LL.
3. Для каждой отдельной модели l=1..Ll=1..L выбирается число признаков dl(dl<D)dl(dl<D). Как правило, для всех моделей используется только одно значение dldl.
4. Для каждой отдельной модели ll создается обучающая выборка путем отбора dldl признаков из DD.
5. Производится обучение моделей l=1..Ll=1..L, каждая модель обучается на отдельном подмножестве из dldl признаков.
6. Чтобы применить модель ансамбля к тестовой выборке, объединяются результаты отдельных моделей или мажоритарным голосованием или более сложными способами.

Почему именно деревья активно используются в ансамблевых моделях? Потому что они очень чувствительны к выбросам и изменению данных.

Незначительное изменение обучающей выборки, удаление или добавление признаков может сильно изменить вид решающего дерева. Это является недостатком с точки зрения построения отдельного решающего дерева. Это же является преимуществом при использовании деревьев в ансамблевой модели, так как с помощью незначительного изменения обучающих данных или набора используемых признаков можно построить очень разные решающие деревья. Различные решающие деревья дадут хорошую "дисперсию" решений при объединении их в ансамбль.

5) Градиентный бустинг

Изначально метод бустинга (от hypothesis boosting - усиление гипотезы) - это любой ансамблевый метод, который способен комбинировать несколько "слабых моделей" в одну "сильную модель".

Сейчас термин "бустинг", как правило, ассоциируется с алгоритмом градиентного бустинга.

В отличие от методов бэггинга и случайного леса, которые ориентированы прежде всего на минимизизацию дисперсии (Variance), методы бустинга ориентированы прежде всего на минимизацию смещения (Bias) и, отчасти, на минимизизацию дисперсии.

Идея градиентного бустинга состоит в том, что строится многослойная модель (классификации или регрессии) и каждый следующий слой пытается минимизировать остаточную ошибку (residual error), допущенную на предыдущем слое.

Каждый следующий слой учится на ошибку (невязку) между предыдущим слоем и истинным значением.

Необходимо отметить, что в случае задачи классификации, для слоев все равно решается задача регресии для вероятностей классов.

11. Формирование обучающей и тестовой выборок.

Обучающая выборка (training sample) — выборка, по которой производится настройка (оптимизация параметров) модели зависимости.

Если модель зависимости построена по обучающей выборке X^m, то оценка качества этой модели, сделанная по той же выборке X^m, оказывается,

как правило, оптимистически смещённой. Это нежелательное явление называют переобучением. На практике оно встречается очень часто. Хорошую эмпирическую оценку качества построенной модели даёт её проверка на независимых данных, которые не использовались для обучения.

Тестовая (или контрольная) выборка (test sample) — выборка, по которой оценивается качество построенной модели. Если обучающая и тестовая выборки независимы, то оценка, сделанная по тестовой выборке, является несмещённой.

В данной работе для разделения выборки на обучающую и тестовую использовали функцию train_test_split. Как правило, параметр test_size устанавливают в 20% или 30%. Здесь используется test_size=0.3 (30%).

Параметр random_state позволяет задавать базовое значение для генератора случайных чисел. Это делает разбиение неслучайным. Если задается параметр random_state то результаты разбиения будут одинаковыми при различных запусках. На практике этот параметр удобно использовать для создания "устойчивых" учебных примеров, которые выдают одинаковый результат при различных запусках.

12.Балансировка данных.

При использовании алгоритма машинного обучения очень важно обучить модель на наборе данных с почти таким же количеством выборок. Это называется сбалансированным классом. Нам нужны сбалансированные классы для обучения модели, но, если классы не сбалансированы, нам нужно использовать метод балансировки классов перед использованием алгоритма машинного обучения.

Так как при проведении разведочного анализа данных был выявлен довольно сильный дисбаланс классов целевого признака stroke (класс 0 составляет 95,13%, а класс 1 составляет 4,87%), то необходимо сбалансировать датасет. Для этого был выбран метод SMOTE.

Передискретизация SMOTE – один из самых надежных подходов, который призван предотвратить дисбаланс классов. Он расшифровывается как

«Техника передискретизации синтетического меньшинства». Этот метод разработан специально для создания новых образцов, соответствующих второстепенным классам.

Принцип работы метода:

- Для произвольного объекта выбираются ближайшие соседи (как правило, K=5).
- Для каждой пары объект-сосед генерируется новый объект на основе формулы:

новый объект=объект−масштаб·(объект−сосед)

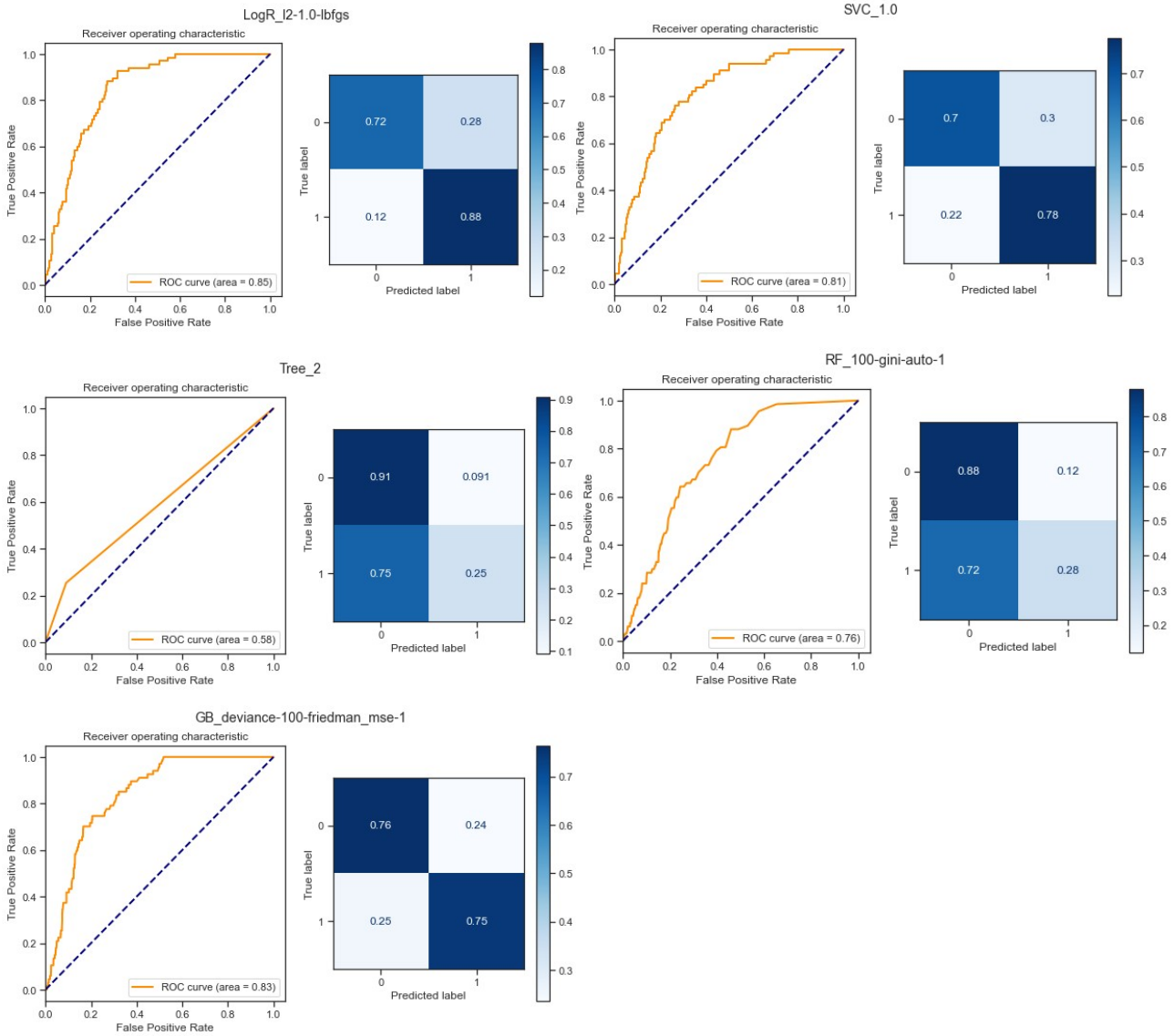
В результате, получено равномерное распределение классов (по 3394 записей каждого класса 0 и 1).

13.Построение базового решения (baseline) для выбранных моделей.

Для построения базового решения использованы классы из библиотеки sklearn:

1) Логистическая регрессия «LogR_l2-1.0-lbfgs» (в названии указаны значения гиперпараметров penalty, C, solver по умолчанию) - LogisticRegression(random_state=3).
2) Машина опорных векторов «SVC_1.0» (в названии указано значение гиперпараметра C по умолчанию) - SVC(probability=True, random_state=3).
3) Решающее дерево «Tree_2» (в названии указано значение гиперпараметра min_samples_leaf по умолчанию) - DecisionTreeClassifier(random_state=3).
4) Случайный лес «RF_100-gini-auto-1» (в названии указаны значения гиперпараметров n_estimators, criterion, max_features, min_samples_leaf по умолчанию) - RandomForestClassifier(random_state=3).
5) Градиентный бустинг «GB_deviance-100-friedman_mse-1» (в названии указаны значения гиперпараметров loss, n_estimators, criterion, min_samples_leaf по умолчанию) - GradientBoostingClassifier(random_state=3).

Получены следующие результаты:



14.Подбор гиперпараметров для выбранных моделей.

Для того чтобы найти комбинацию гиперпараметров, которая будет лучшей для модели, был использован метод Grid Search (поиск по решетке, решетчатый поиск).

Перекрестная проверка — метод оценки аналитической модели и её поведения на независимых данных. При оценке модели имеющиеся в наличии данные разбиваются на k частей. Затем на k−1 частях данных производится обучение модели, а оставшаяся часть данных используется для тестирования. Процедура повторяется k раз; в итоге каждая из k частей данных используется для тестирования. В результате получается оценка эффективности выбранной модели с наиболее равномерным использованием имеющихся данных.

В результате, подобраны следующие гиперпараметры:

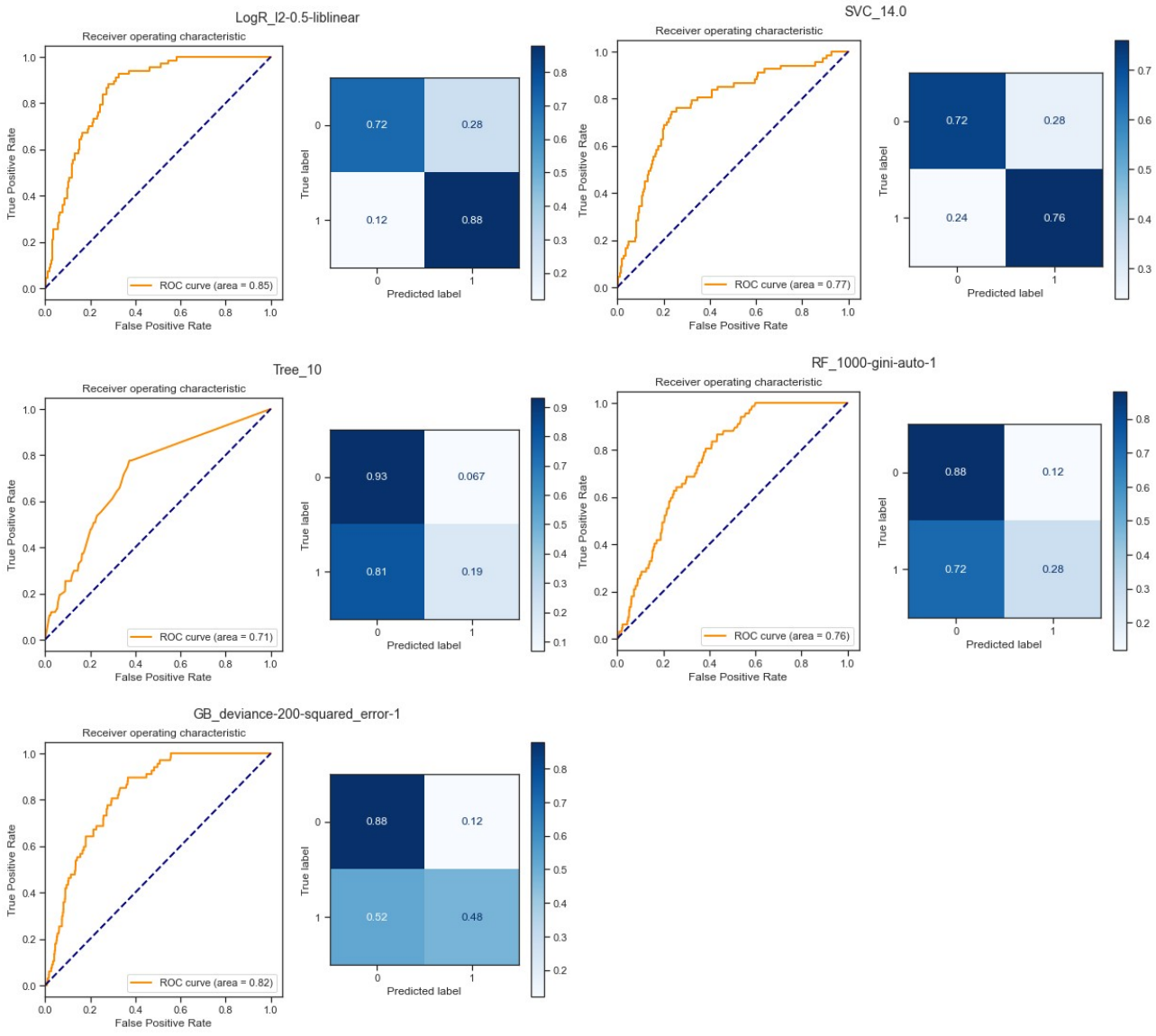
1) Для модели логистической регрессии: C=0.5, solver='liblinear', penalty='l2'.
2) Для модели машины опорных векторов: C=14.0.

3) Для модели решающего дерева: min_samples_leaf=10.
4) Для модели случайного леса: n_estimators=1000, criterion='gini', max_features='auto', min_samples_leaf=1.
5) Для модели градиентного бустинга: loss='deviance', n_estimators=200, criterion='squared_error', min_samples_leaf=1.

15. Построение решения для найденных оптимальных значений гиперпараметров.

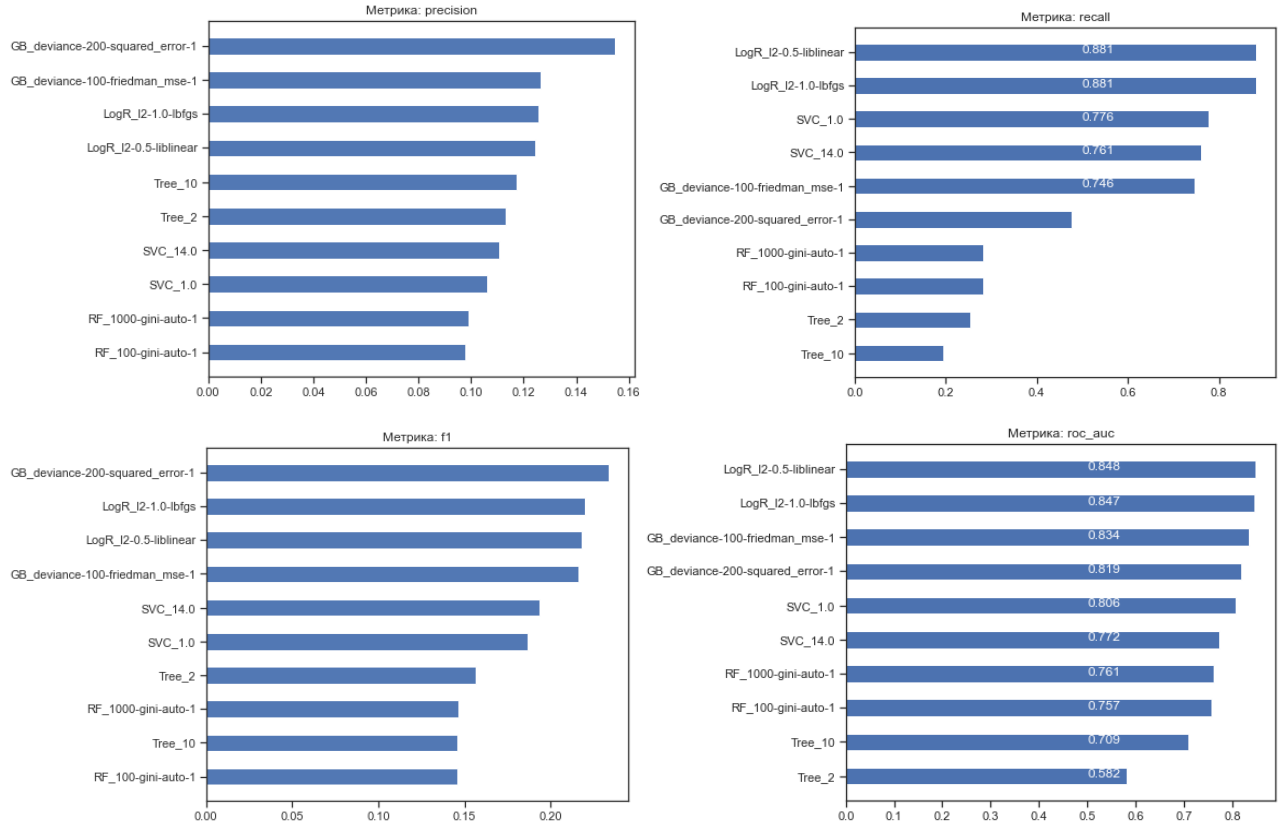
Для построения решения для найденных оптимальных значений гиперпараметров были использованы классы из библиотеки sklearn, которые также были использованы для построения базового решения, только с указанием значений подобранных гиперпараметров.

Получены следующие результаты:



16. Сравнение качества оптимальных моделей с качеством baseline-моделей. Выводы о качестве построенных моделей на основе метрик.

Для того чтобы сравнить качество построенных моделей, были построены графики метрик качества моделей (precision, recall, f1, roc_auc):



По графикам видно, что не на всех метриках оптимальные модели показывают лучшее качество по сравнению с baseline-моделями. Такое происходит, так как кросс-валидация максимизиует метрику не для тестовой выборки, а по очереди для n блоков (cv=n) из тренировочной, что в теории даёт больше уверенности, что модель будет лучше работать и на тесте. Но это не гарантирует, что модель на любом тесте будет лучше. В долгосрочной перспективе, если бы подавалось большое количество тестовых данных, то подобранная модель опередила бы базовую.

На основании двух метрик (recall, roc_auc) из четырех используемых, лучшей оказалась модель логистической регрессии. На основании двух других метрик (precision, f1) лучшей оказалась модель градиентного бустинга. Как видно по графикам, эти модели могут становиться лучше или хуже по сравнению друг с другом в зависимости от тестовой выборки, так как разрыв между их показателями небольшой. Таким образом, обе модели, логистическая регрессия и градиентный бустинг, показали примерно равные результаты и являются лучшими.

17.Заключение

В результате проделанной работы была решена задача машинного обучения. Построены пять моделей машинного обучения, в том числе ансамблевые, для решения задачи бинарной классификации: логистическая регрессия, машина опорных векторов, решающее дерево, случайный лес, градиентный бустинг. Построено базовое решение для выбранных моделей без подбора гиперпараметров и решение для найденных оптимальных значений гиперпараметров. Проведено сравнение качества полученных оптимальных моделей с качеством baseline-моделей. Сделаны выводы по качеству построенных моделей на основе четырех метрик: precision, recall, f1, roc_auc.

18.Список литературы

1) https://scikit-learn.org/stable/
2) https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
3) https://medium.com/analytics-vidhya/derivative-of-log-loss-function-for-logistic-regression-9b832f025c2d
4) Дж. Вандер Плас. Python для сложных задач. // ПИТЕР. – 2018.
5) Крис Элборн. Машинное обучение с использованием Python. Сборник рецептов. // БХВ-Петербург. – 2019.