

КАФЕДРА Системы обработки информации и управления

Прогнозирование продолжения обучения в университете

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 продолжения обучения в университете

Студент группы РТ5-61Б

Евсеев Георгий Андреевич
(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

(Подпись, дата) Гапанюк Ю.Е.
(И.О.Фамилия)

Студент

(Подпись, дата) Евсеев Г.А.
(И.О.Фамилия)

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

Содержание

1. Введение
2. Описание набора данных
3. Постановка задачи
4. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных.
5. Кодирование категориальных признаков.
6. Масштабирование данных.
7. Проведение корреляционного анализа данных. Выбор признаков, подходящих для построения моделей.
8. Выбор метрик.
9. Выбор моделей машинного обучения.
10. Формирование обучающей и тестовой выборок.
11. Построение базового решения (baseline) для выбранных моделей.
12. Подбор гиперпараметров для выбранных моделей.
13. Построение решения для найденных оптимальных значений гиперпараметров.
14. Сравнение качества оптимальных моделей с качеством baseline-моделей.
Выводы о качестве построенных моделей на основе метрик.
15. Заключение
16. Список литературы

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 «Go To College Dataset» со следующими признаками:

type_school - тип школы, которую посещал студент (Academic, Vocational).

school_accreditation - качество школы (A, B).

gender - Пол студента (Male, Female).

interest - заинтересованность студента в поступлении в университет(very, less, uncertain, quite, not).

residence - местность проживания(rural - сельская, urban - городская).

parent_age - возраст родителей.

parent_salary - Заработок родителей.

house_area - площадь родительского дома.

average_grades - средняя оценка в школе.

parent_was_in_college - были ли родители студента в университете(True, False).

in_college – целевой признак датасета: поступил ли студент в университет(True, False).

3. Постановка задачи

В данной работе будет решаться задача бинарной классификации на примере прогнозирования вероятности продолжения обучения студента.

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 `in_college`. 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.

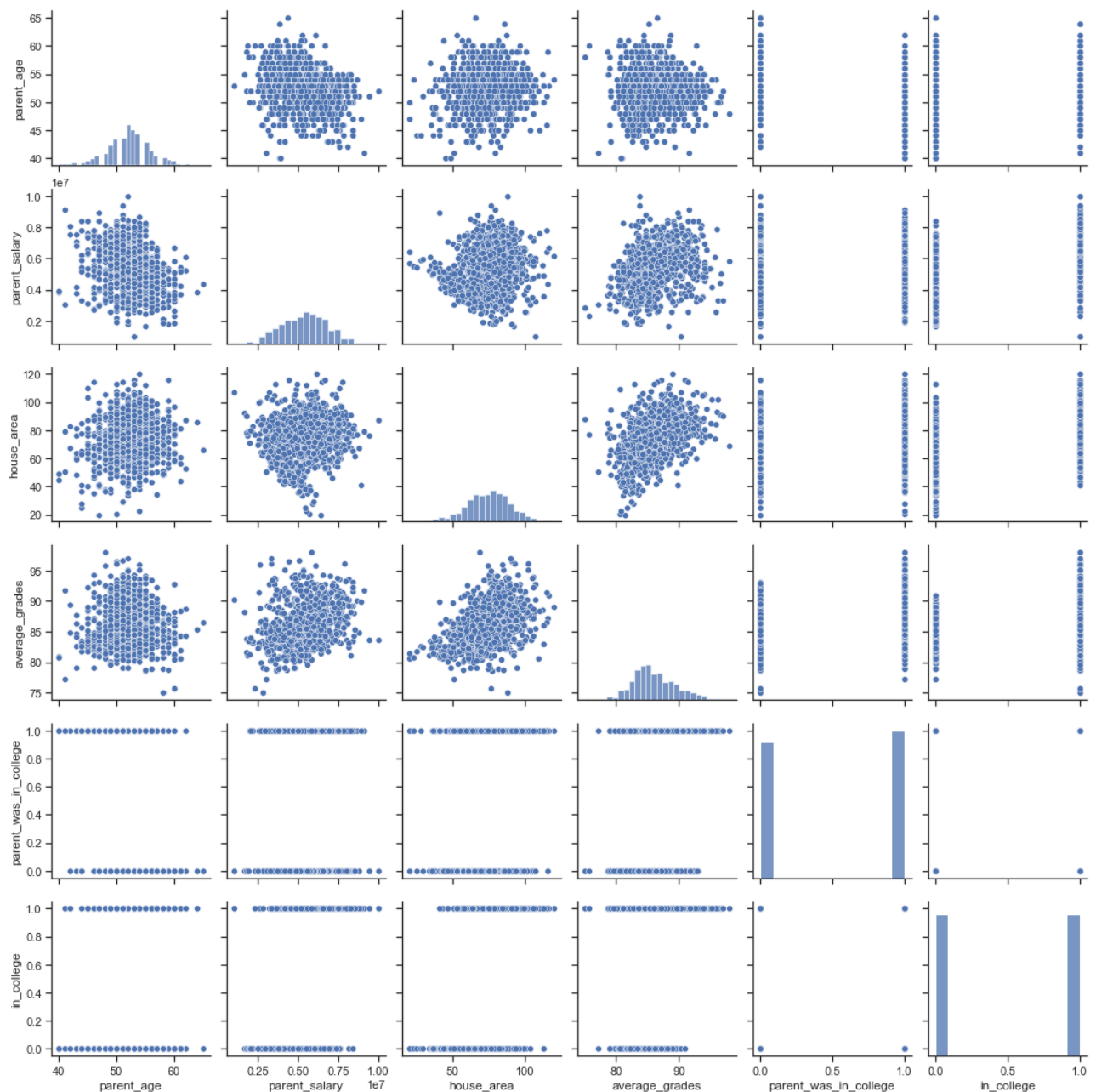
Разведочный анализ данных (Exploratory Data Analysis) – предварительное исследование датасета с целью определения его основных характеристик, взаимосвязей между признаками, а также сужения набора методов, используемых для создания модели машинного обучения.

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

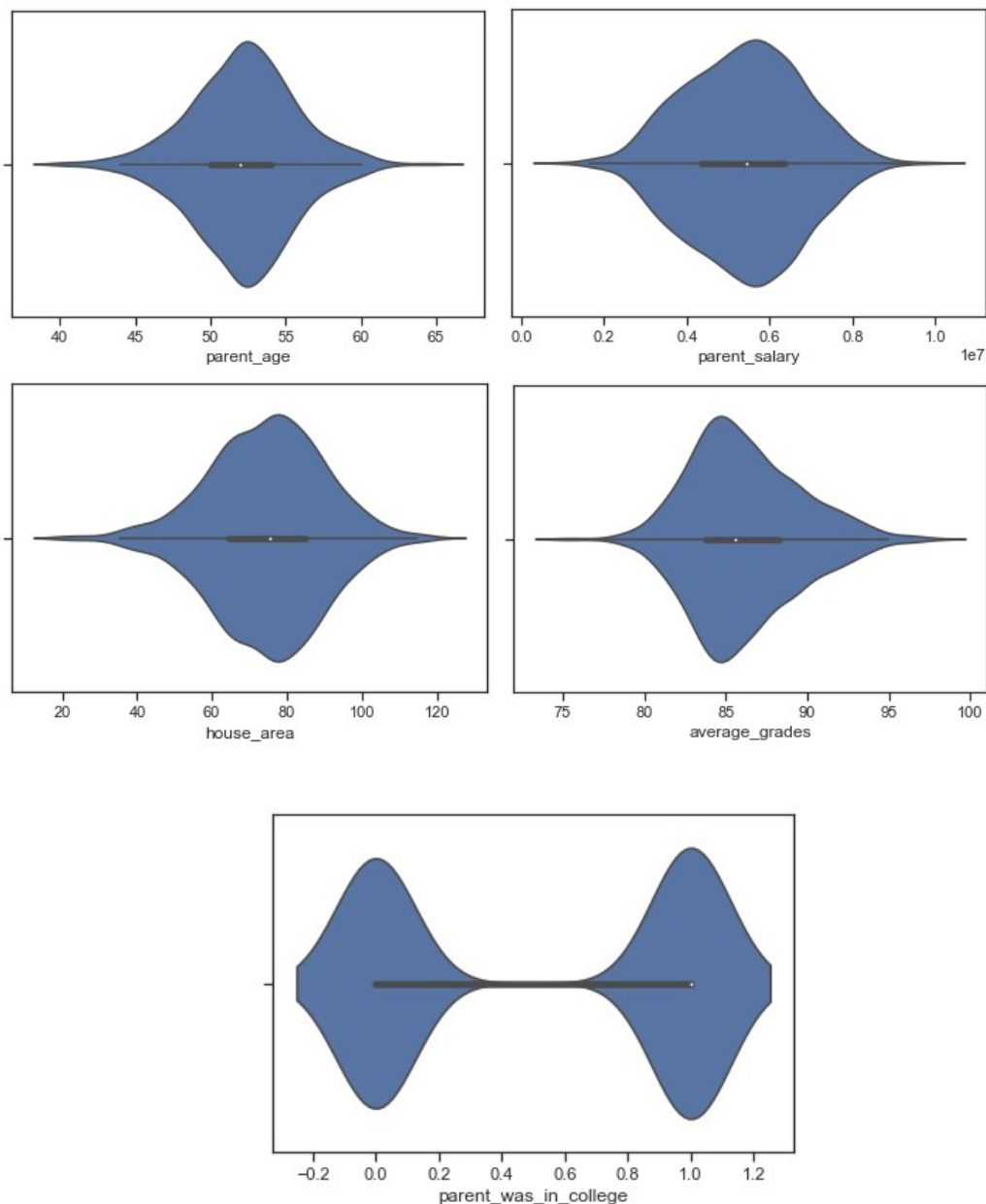
- 1) После удаления дубликатов записей, размер датасета - 1000 строк, 11 колонок.

- 2) В датасете представлены данные следующих типов: int64, object, float64, boolean.
- 3) Датасет не содержит пропусков, так что обработка пропусков не требуется.
- 4) Целевой признак in_college для задачи бинарной классификации, действительно, содержит только 0 и 1.
- 5) Дисбаланс классов категориального признака отсутствует.

Также были построены парные диаграммы для понимания структуры данных:



Были построены скрипичные диаграммы для числовых колонок (parent_age, parent_salary, house_area, average_grades, parent_was_in_college):



5. 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: `type_school`, `school_accreditation`, `gender`, `interest`, `residence`. Данные признаки содержат следующие классы:

- 1) Уникальные значения признака `type_school`: ['Academic' 'Vocational']
- 2) Уникальные значения признака `school_accreditation`: ['A' 'B']
- 3) Уникальные значения признака `gender`: ['Male' 'Female']
- 4) Уникальные значения признака `residence`: ['Urban' 'Rural']
- 5) Уникальные значения признака `interest`: ['less interested' 'Very interested' 'Uncertain' 'not interested', 'quiet interested']

После кодирования категориальных признаков в датасете остались данные следующих типов: `int32`, `float64`, `int64`.

6. 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:

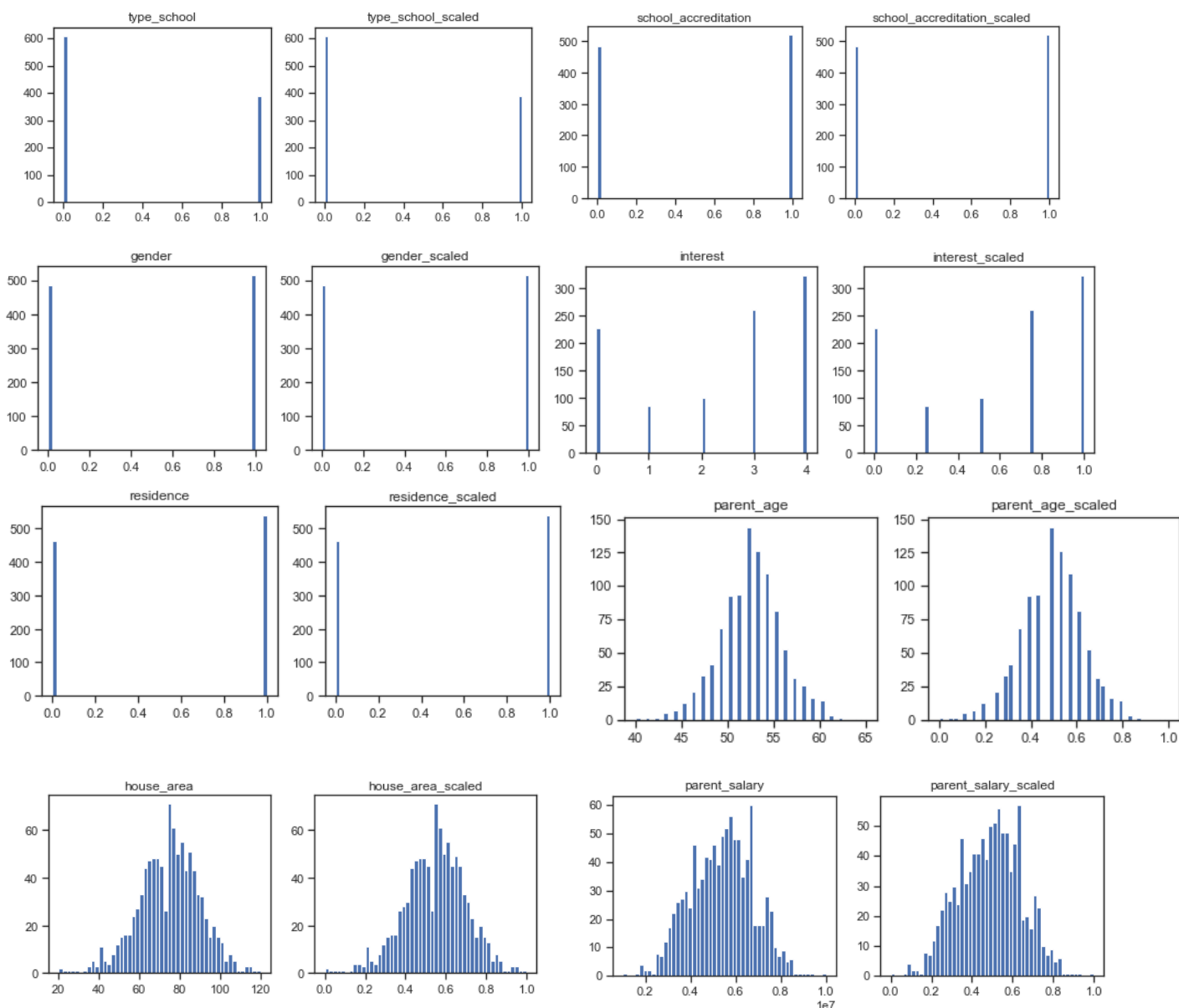
$$x_{\text{новый}} = \frac{x_{\text{старый}} - \text{AVG}(X)}{\sigma(X)}$$

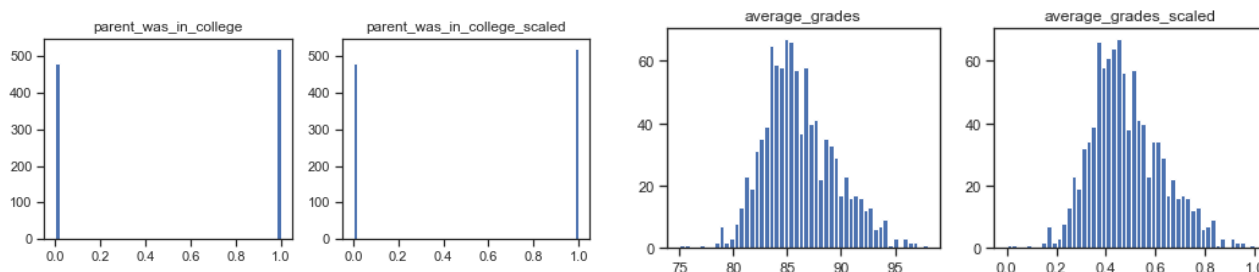
В этом случае большинство значений попадает в диапазон от -3 до 3.

где X - матрица объект-признак, $\text{AVG}(X)$ - среднее значение, σ - среднеквадратичное отклонение.

В данной работе использован метод MinMaxScaler из библиотеки sklearn. Были масштабированы все колонки.

Убедились, что масштабирование не повлияло на распределение данных:





7. 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.

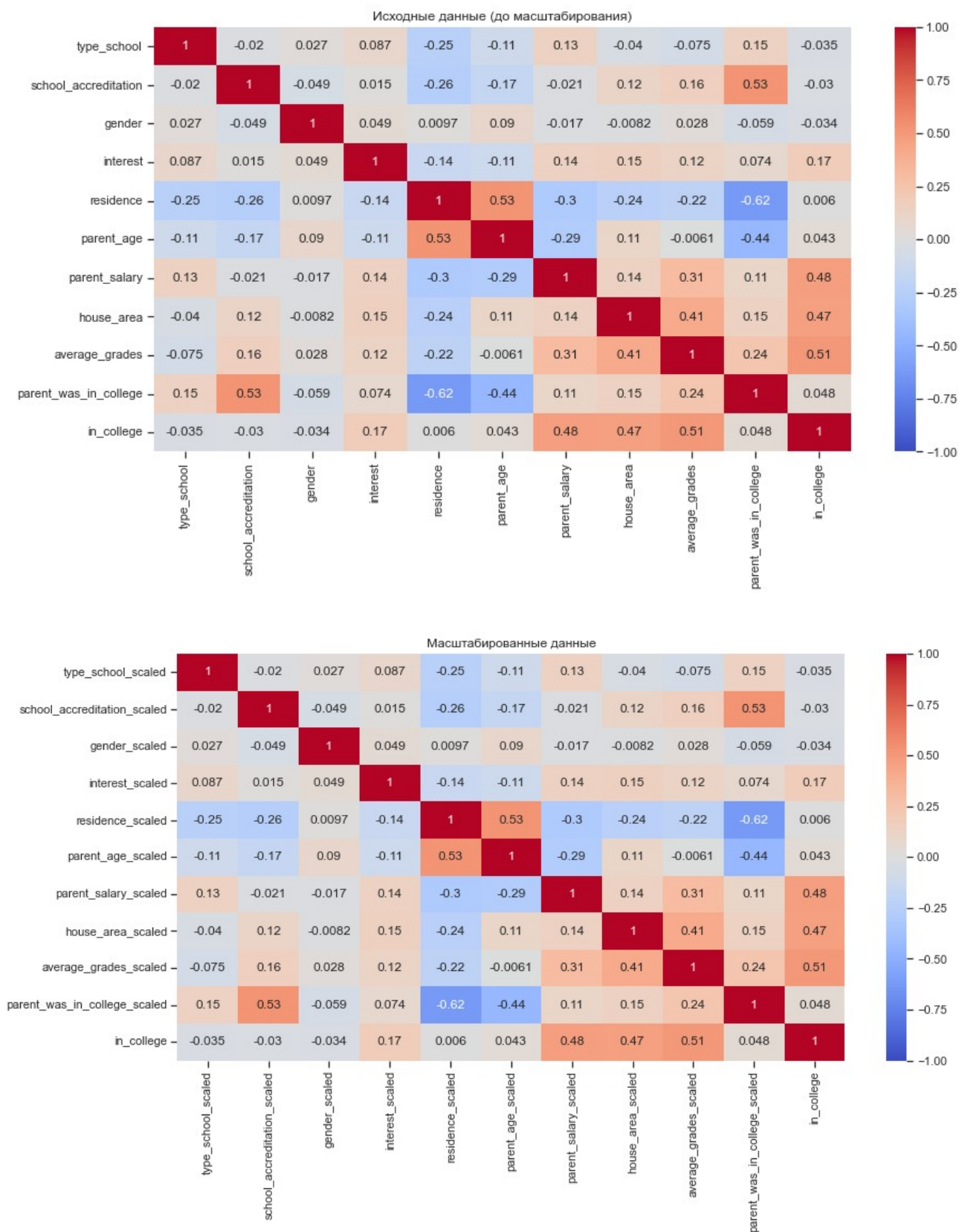
Коэффициент корреляции — двумерная описательная статистика, количественная мера взаимосвязи (совместной изменчивости) двух переменных.

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.

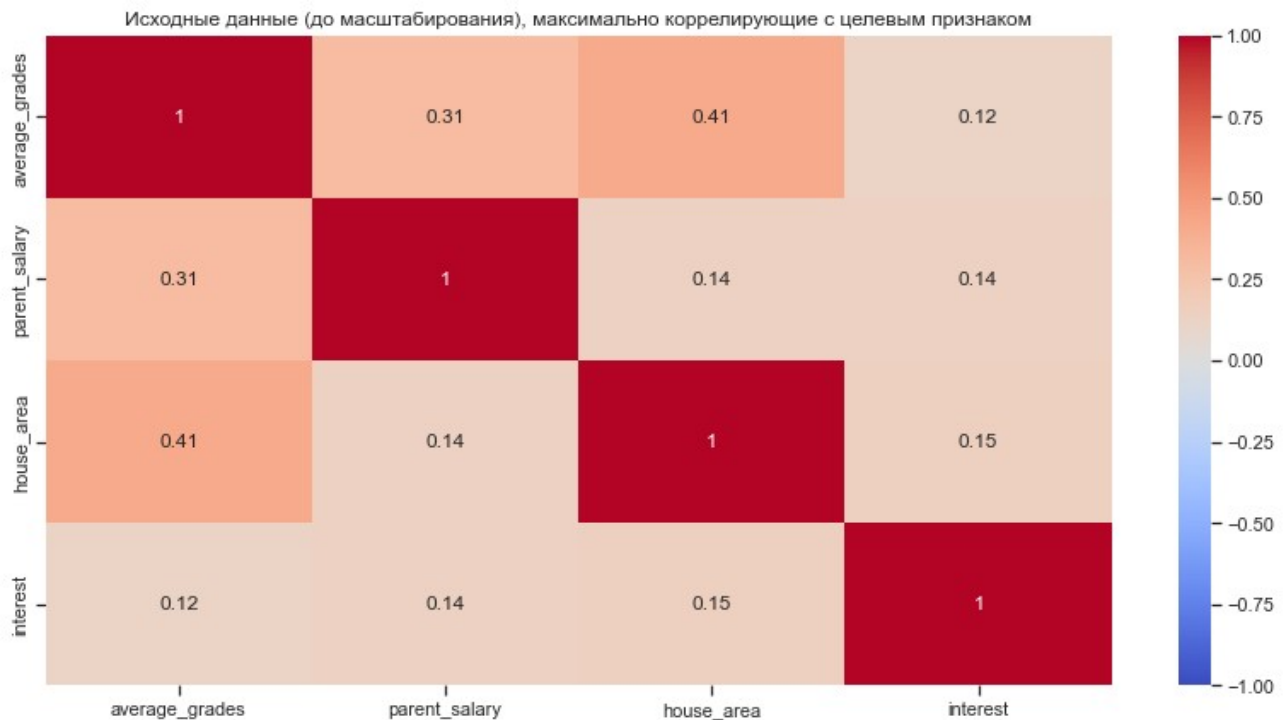
При положительной линейной корреляции более высоким значениям одного признака соответствуют более высокие значения другого, а более низким значениям одного признака — низкие значения другого.

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): `average_grades`, `parent_salary`, `house_area`, `interest`. 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 `in_college` наиболее сильно коррелирует с средними оценками в школе студента (0.50), заработной платой родителей (0.47), площадью дома (0.46), заинтересованностью в университете (0.17). Эти признаки обязательно следует оставить в модели классификации.

8. 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 **F1**-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 $\beta=1$:

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

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`.

9. 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

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

(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:

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

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:

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

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

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

Частные производные полной функции потерь по коэффициентам регрессии 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 (x_1, x_2)

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 $l=1..L$ выбирается число признаков dl ($dl < D$). Как правило, для всех моделей используется только одно значение $dldl$.
4. Для каждой отдельной модели l создается обучающая выборка путем отбора $dldl$ признаков из DD .
5. Производится обучение моделей $l=1..L$, 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 минимизацию дисперсии (Variance), методы бустинга ориентированы прежде всего на минимизацию смещения (Bias) и, отчасти, на минимизацию 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 регрессии для вероятностей классов.

10. 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%).

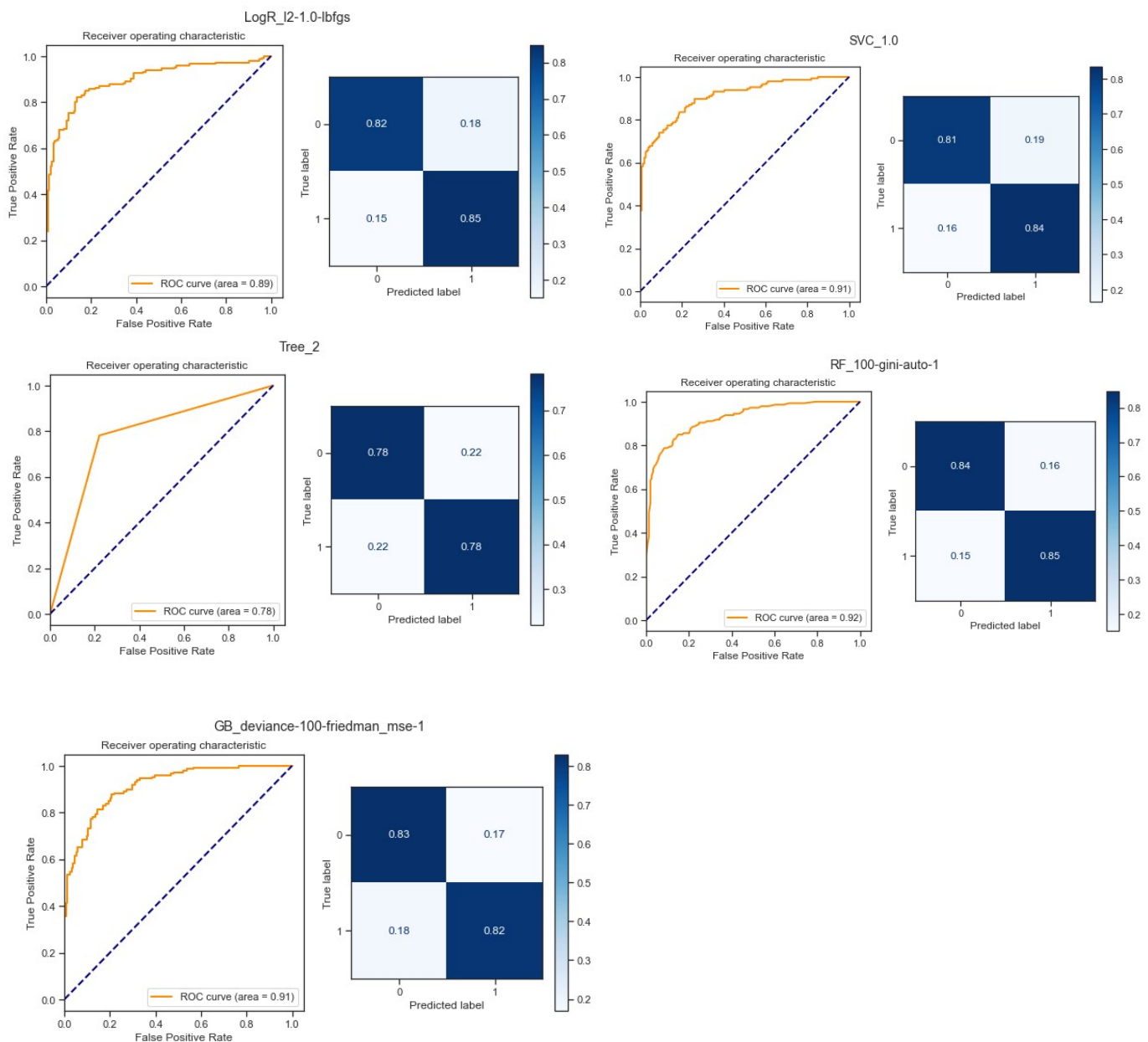
Параметр `random_state` позволяет задавать базовое значение для генератора случайных чисел. Это делает разбиение неслучайным. Если задается параметр `random_state` то результаты разбиения будут одинаковыми при различных запусках. На практике этот параметр удобно использовать для создания "устойчивых"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

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

Для построения базового решения использованы классы из библиотеки `sklearn`:

- 1) Логистическая регрессия «LogR_12-1.0-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:



12. 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.

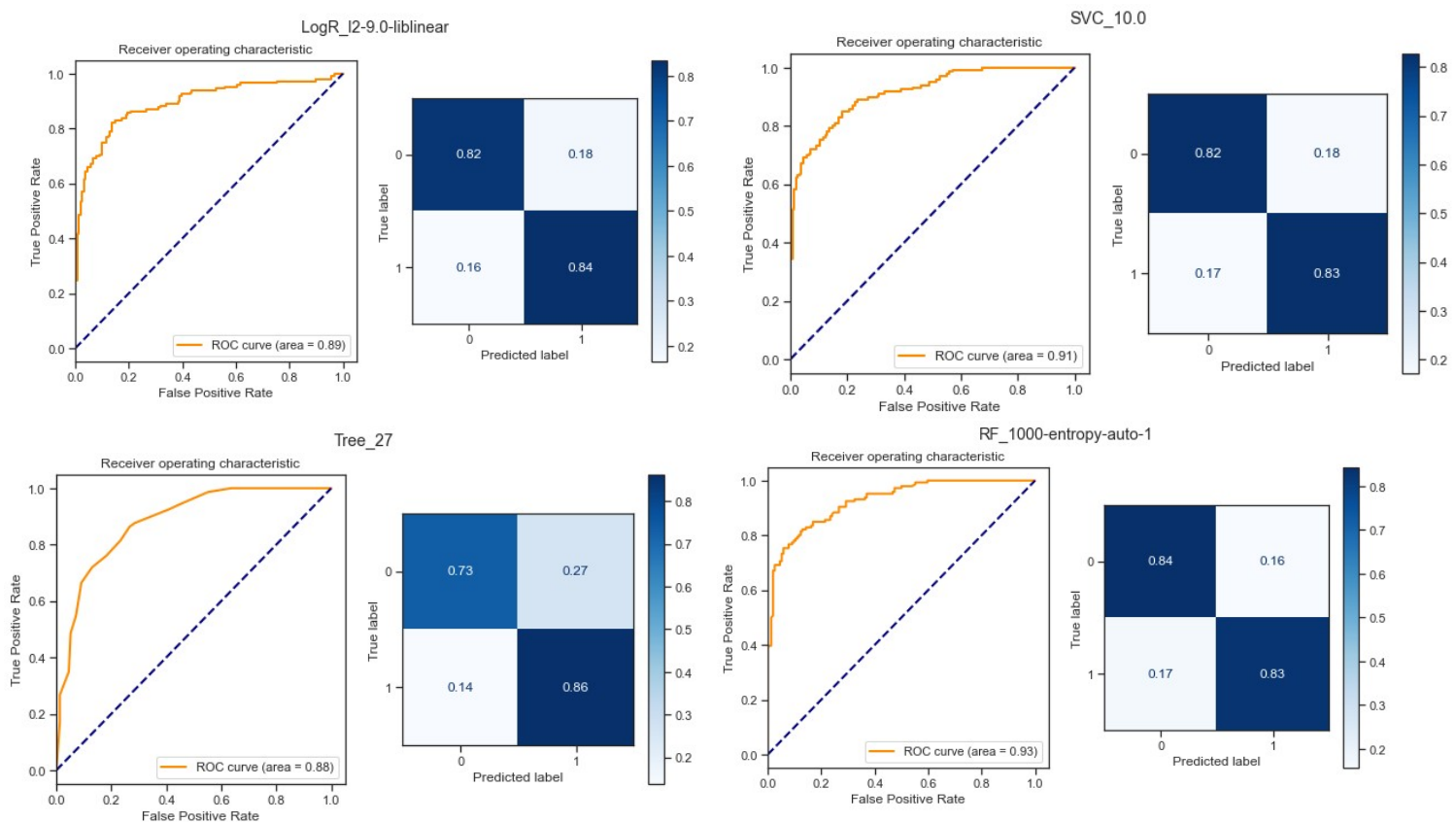
В результате, подобраны следующие гиперпараметры:

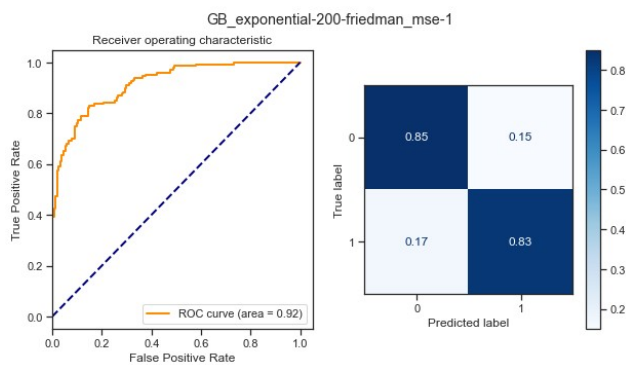
- 1) Для модели логистической регрессии: $C=9.0$, `solver='liblinear'`, `penalty='l2'`.
- 2) Для модели машины опорных векторов: $C=10.0$.
- 3) Для модели решающего дерева: `min_samples_leaf=27`.
- 4) Для модели случайного леса: `n_estimators=1000`, `criterion='entropy'`, `max_features='auto'`, `min_samples_leaf=1`.
- 5) Для модели градиентного бустинга: `loss='exponential'`, `n_estimators=200`, `criterion='friedman_mse'`, `min_samples_leaf=1`.

13. 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.

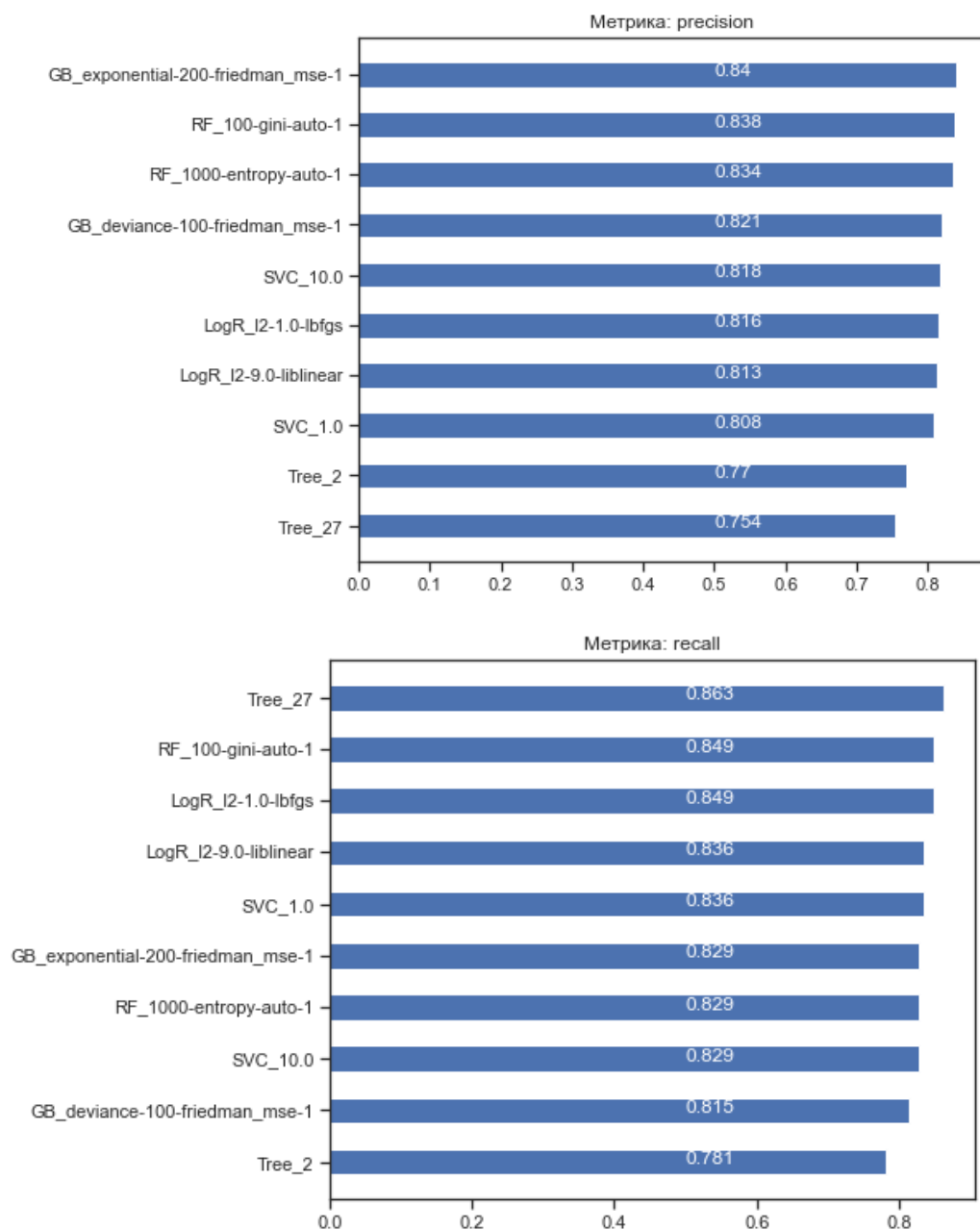
Получены следующие результаты:

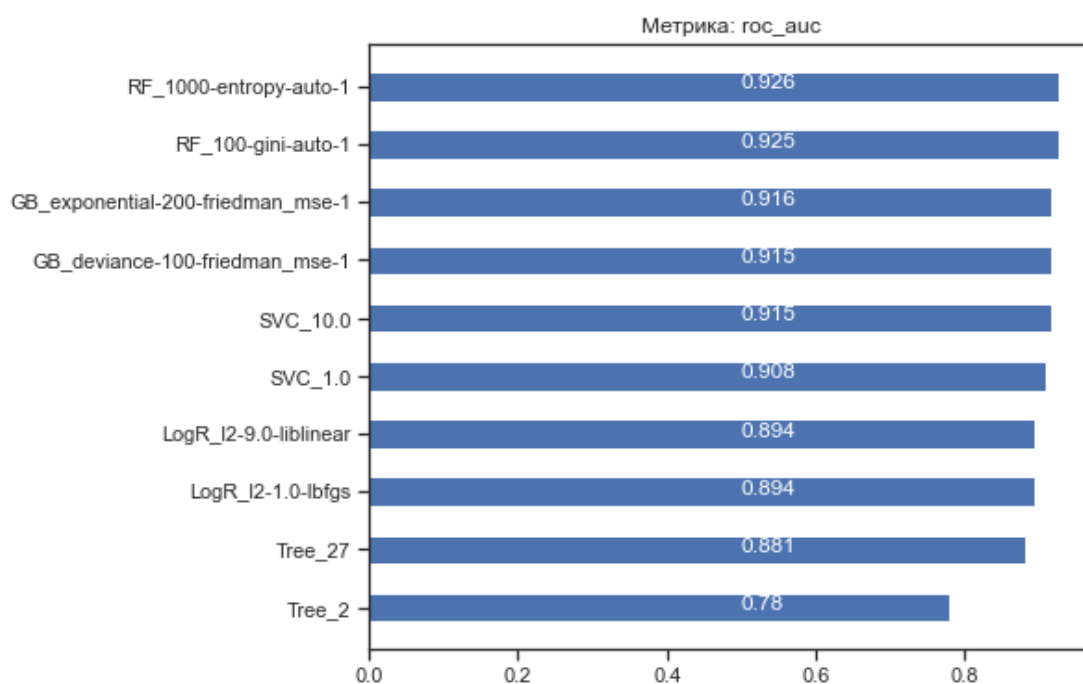
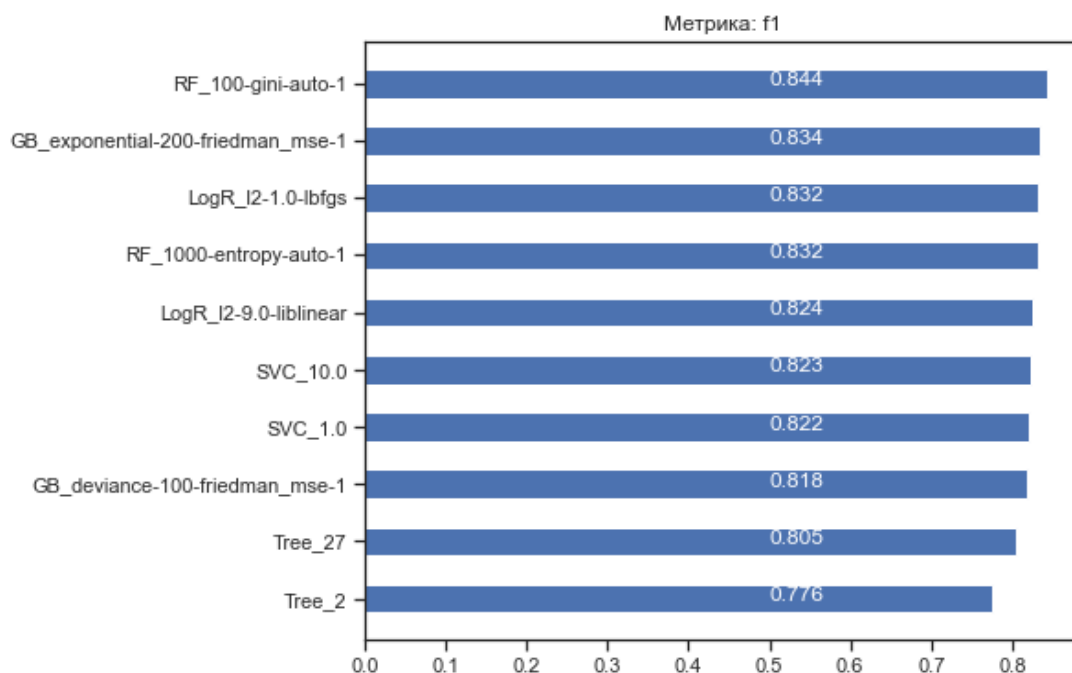




14. Сравнение качества оптимальных моделей с качеством baseline-моделей. Выводы о качестве построенных моделей на основе метрик.

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 максимизирует 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 метрики точности, лучшей оказалась модель градиентного бустинга. На основании метрики recall лучшей оказалась модель решающего дерева. На основании метрики f1, лучшей оказалась модель случайного дерева, и на основании метрики roc_auc лучшей моделью оказалась также модель случайного дерева. Таким образом, все модели показывают высокие результаты в разных метриках, но более оптимальной является модель случайного дерева на основании двух метрик(f1,roc_auc).

15.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.

16.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.