Аннотация.

В наше время большую часть работы по принятию решений в финансовой сфере доверяют вычислять компьютерам. Кредитный скоринг не стал исключением. Существует множество методов подбора гиперпараметров для данной задачи, однако не выявлено лучшего, поэтому проводится множество сравнений моделей, чтобы выявить в каком случае использовать тот или иной метод. В данной статье изучаются четыре метода подбора гиперпараметров Grid Search, Random Search, Баесовская Оптимизация, Генетический алгоритм для разных моделей классификации. Сравним метрики и время обучения для методов и моделей. Данные методы выбраны из-за их высокой популярности и частой практической используемости, над ними проводилось множество исследований, но вместе их не сравнивали. Результаты экспериментов показали, что результат работа методов почти аналогичен, но за наименьшее время выполнил свою работу Random Search – 5 сек, в то время как Grid Search – 13 сек, Bayes Search –75 сек., Генетический алгоритм – 316 сек.

# Введение

## Актуальность

Искусственный интеллект сегодня развивается семимильными шагами, предлагая огромную помощь людям во всех сферах бизнеса и экономики. ИИ — это общий термин, описывающий множество машин и алгоритмов, имитирующих когнитивные функции человека. С помощью ИИ компьютерные решения принимаются намного быстрее из-за значительно более высокой скорости обработки данных компьютерными системами.

Кредитный скоринг на основе ИИ, пожалуй, самый перспективный и актуальный. В двух словах, кредитный скоринг представляет собой оценку того, насколько хорошо клиент банка может платить и готов погасить долг. Учитывая, что во всем мире 2,5 миллиарда человек не пользуются банковскими услугами, и только менее половины населения, пользующегося банковскими услугами, считается имеющим право на получение кредита, потребность в более разумных решениях для оценки кредитоспособности очевидна. Решения по оценке кредитоспособности ИИ основаны на большом количестве данных, таких как общий доход, кредитная история, анализ транзакций, опыт работы и даже Google Analytics. По сути, скоринг представляет собой математическую модель, основанную на статистических методах и учитывающую большой объем информации. В результате кредитный скоринг с использованием ИИ обеспечивает более точную, индивидуализированную оценку кредитного рейтинга на основе множества дополнительных факторов в реальном времени, предоставляя доступ к финансированию большему количеству людей с потенциальным доходом.

В большинстве финансовых организаций модели кредитного скоринга до сих пор основаны на скоринговом подходе, т. е. на динамике, характерной для времени их создания. Потенциальный заемщик должен обладать достаточными историческими данными о предыдущем поведении заимствования, чтобы быть оцененным как «поддающийся оценке». При отсутствии такой исторической информации (что является типичной ситуацией для новых клиентов банковского сектора) даже кредитоспособным заемщикам отказывают в доступе к кредиту. В отличие от традиционных методов кредитного скоринга (например, метод оценочной карты), которые фокусируются на прошлой деятельности заемщика, кредитный скоринг ИИ более чувствителен к показателям кредитоспособности потенциального заемщика в режиме реального времени, таким как текущий уровень дохода, возможности трудоустройства и их потенциальная способность зарабатывать. Таким образом, заемщики с высоким потенциалом включаются в кредитные программы, а те, кто формально проходит традиционный кредитный скоринг (например, сбрасыватели кредитных карт), из них исключаются. Другими словами, кредитный скоринг на основе ИИ позволяет точно прогнозировать прибыль на основе интеллектуальных моделей ИИ.

## Цель

Цель работы состоит в том, чтобы сравнить четыре метода подбора гиперпараметров Grid Search, Random Search, Баесовская Оптимизация, Генетический алгоритм и на основе результата исследований использовать лучший из них для данной задачи. В ходе исследования эти модели будут обучаться на одних и тех же данных и сравниваться между собой метриками. Они были выбраны, поскольку на данный момент эти четыре подхода являются одними из самых популярных и не сравнивались друг с другом. В частности, какой алгоритм покажет наилучшие результаты и что будет способствовать этому.

## Содержание

В разделе 2 рассматриваются существующие работы по подбору гиперпараметров для моделей, решающих задачу классификации. В разделе 3 описано используемое оборудование и технологии. В разделе 4 описание набора данных. В разделе 5 описание методов подбора гиперпараметров и алгоритмов классификации. В 6 разделе сравнение результатов. В разделе 7 выводы.

# Обзор работ

Производительность модели тесно связана с настройкой гиперпараметров[7]. Гиперпараметры могут определять сложность модели. Например, параметры скрытого слоя могут определять количество каскадных слоев свертки; гиперпараметры могут определять эффективность обучения и конечную производительность модели, например, скорость обучения и размер пакета.

В [1] авторы указывают на важность концепции досрочного завершения, которая является хорошим способом оптимизации гиперпараметров, поскольку нормальный процесс оптимизации требует больших вычислительных мощностей и времени из-за количества гиперпараметров, требуемых одновременно, который может быть длинным; до определенного значения. диапазон значений или даже несколько значений. Вместо этого основная идея предлагаемого подхода состоит в том, чтобы обучать каждую модель в течение короткого периода времени, а затем извлекать доступную информацию для прогнозирования окончательного результата модели. Модель была проверена путем тестирования банковской маркетинговой базы данных с 45 211 записями и 17 атрибутами. Результаты сравнивались с использованием случайного поиска и алгоритма Predictive Hyperparameter Optimization (PHO) (их предложенная модель). Все результаты показали, что PHO работает лучше, чем случайный поиск.

В [2] сравнивались поиск по сетке и поиск вручную; Это название дается алгоритмами при расчете гиперпараметров. Вывод статьи заключался в том, что случайный поиск дает лучшие результаты, чем поиск по сетке, для процесса настройки гиперпараметров как эмпирически, так и теоретически. Эмпирические данные основаны на крупных исследованиях с использованием поиска по сетке и ручного поиска в глубоком убеждении и нейронных сетях. По сравнению с нейронными сетями с настройкой поиска по сетке случайный поиск в той же области может найти такие же или лучшие модели за долю того же вычислительного времени. При том же вычислительном бюджете, что и при случайном поиске, случайный поиск находит лучшие модели, подходящие для поиска в большем пространстве.

В [5] представили структуру прогнозирующего мониторинга бизнес-процессов, основанную на двух методах оптимизации гиперпараметров. Предлагаемые методы основаны на генетических алгоритмах для масштабируемого поиска в пространстве гиперпараметров с целью выявления конфигураций, которые либо максимизируют одноцелевую функцию, либо вычисляют фронты Парето, из которых пользователь может выбрать данную конфигурацию в зависимости от желаемых компромиссов. В частности, показали, как генетические алгоритмы можно успешно использовать для автоматической настройки методов для решения двух комбинированных подзадач: кластеризации (необходимой для извлечения информации с точки зрения потока управления выполнения бизнес-процессов) и классификации (для получения информации из данных). полезные нагрузки, прикрепленные к каждому событию). Затем продемонстрировали, что настраиваемые генетические алгоритмы позволяют (i) сократить время выполнения для оптимизации гиперпараметров и (ii) найти баланс в оптимизации различных измерений, которые участвуют в задаче прогнозирующего мониторинга (например, точность , своевременность и частота отказов).

Наиболее распространенное использование байесовской оптимизации (BO) заключается в решении задач глобальной оптимизации, где цель связана с функцией черного ящика [3]. В связи с этим в литературе изучен ряд подходов к такого рода глобальной оптимизации [21–24]. Методы стохастической аппроксимации, такие как интервальная оптимизация и методы ветвей и границ, эффективны для оптимизации неизвестных целевых функций в машинном обучении [25]. Поэтому гиперпараметрическая оптимизация [26] в машинном обучении относится к значениям, установленным перед обучением, и БО может служить альтернативой одному из методов оптимизации для автоматического задания этих значений.

В меньшей степени обозревали следующие работы:

* [14], сравнивался генетический алгоритм с Байесовской оптимизацией;
* [15], рассматривался подбор гиперпараметров в области физики;
* [16], использовалась Байесовская оптимизация;
* [17], подбор гиперпараметров методом GridSearch в области кибербезопасности;
* [18], методы поиска гиперпараметров;
* [19], использование Grid Search.

# Используемые оборудование и технологии

Компьютер:

* Процессор - AMD Ryzen 5 2600,
* Оперативной память - 16 GB.

Язык программирования - Python 3.10.5.

Используемые библиотеки:

* scikit-learn
* boruta
* sklearn-genetic-opt
* pandas
* scikit-optimize

Используемые модели:

* SVM
* K Nearest Neighbors
* Random Forest
* Gradient Boosting

Используемые методы:

* Grid Search
* Randomized Search
* Bayes Search
* Genetic Algorithm

# Описание набора данных.

В данной работе будет использоваться набор данных [SF-DST] Credit Scoring [6]. Выбранные данные были проверены сообществом участниками в размере 500 человек. Был выбрал данный набор данных, т.к. содержит данные русских пользователей банков. Набор данных состоит из 73799 строк и 11 полей.

Описания полей:

* client\_id - идентификатор клиента
* education - уровень образования
* sex - пол заемщика
* age - возраст заемщика
* car - флаг наличия автомобиля
* car\_type - флаг автомобиля иномарки
* decline\_app\_cnt - количество отказанных прошлых заявок
* good\_work - флаг наличия “хорошей” работы
* bki\_request\_cnt - количество запросов в БКИ
* home\_address - категоризатор домашнего адреса
* work\_address - категоризатор рабочего адреса
* income - доход заемщика
* foreign\_passport – флаг наличия загранпаспорта
* sna - связь заемщика с клиентами банка
* first\_time - давность наличия информации о заемщике
* score\_bki - скоринговый балл по данным из БКИ
* region\_rating - рейтинг региона
* app\_date - дата подачи заявки
* default - флаг выдачи кредита

Для обучения моделей использовали следующие признаки: количество отказанных прошлых заявок, наличие «хорошей» работы, результаты, пол, наличие машины, наличие заграничного паспорта, возраст, оценка из «Бюро кредитных историй», количество запросов в «Бюро кредитных историй», рейтинг региона, доход заемщика. На рисунке 1 представлены круговые диаграммы, а на рисунке 2 гистограммы для описания соответствующих столбцов.

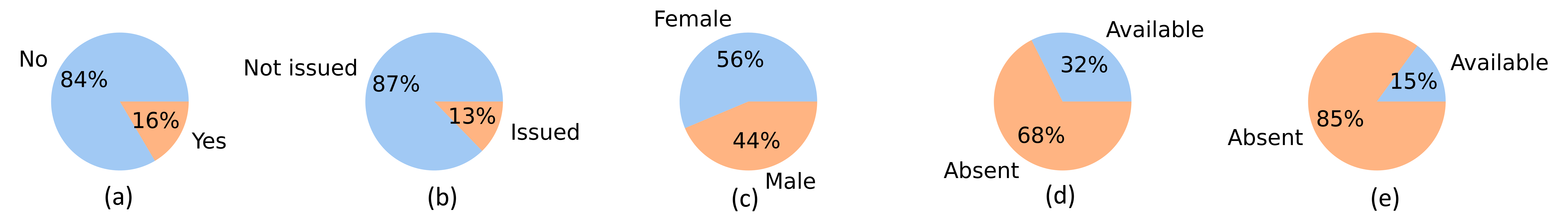


Рисунок 1 – Круговые диаграммы столбцов набора данных (a – наличие «хорошей» работе, b – флаг выдачи кредита, c – пол, d – флаг наличия машины, e – флаг наличия заграничного паспорта)

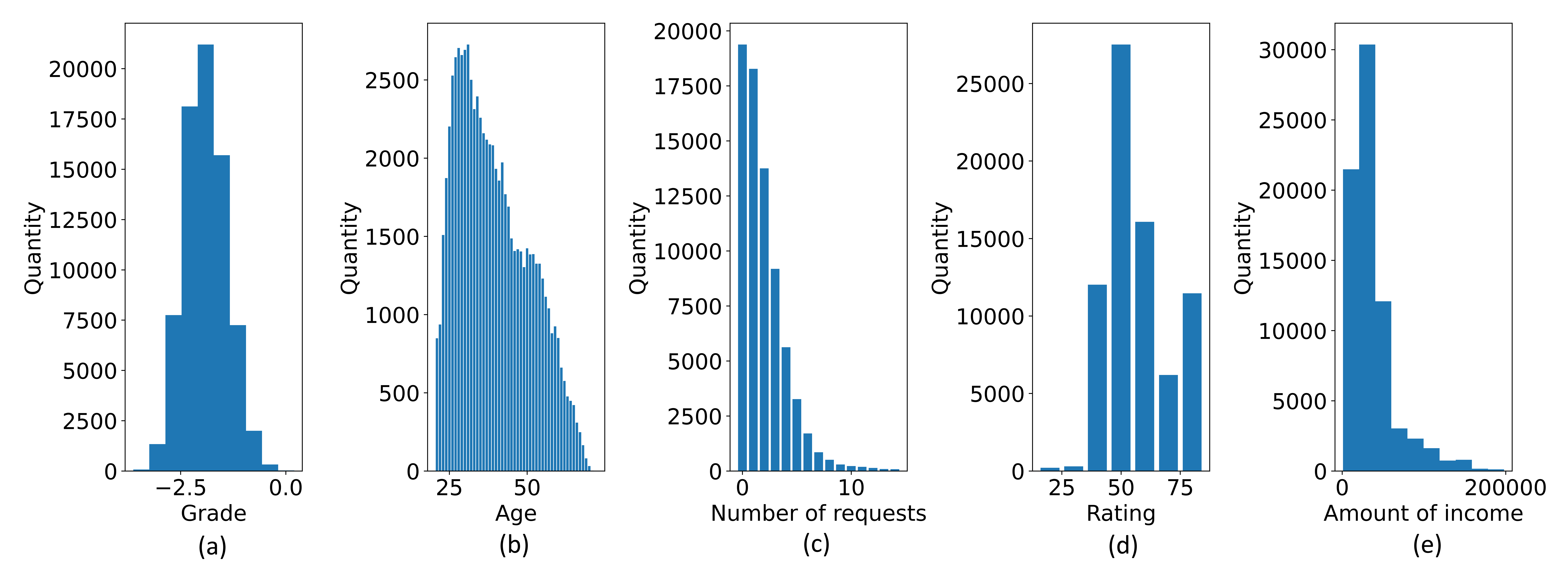


Рисунок 2 – Гистограммы столбцов набора данных (a – скоринговый балл по данным из БКИ, b – возраст, c – количество запросов в БКИ, d – рейтинг региона, e – доход заемщика)

После загрузки данных происходит их обработка. Для каждого числового признака применялась стандартизация, путем приведения данных к отрезку от 0 до 1, а для каждого категориального признака применялся One-Hot Encoding (это тип векторного представления, в котором все элементы в векторе равны 0, за исключением одного, значение которого равно 1, где 1 представляет собой boolean определяющее категорию элемента). Также было произведено удаление выбросов с помощью метода 3-х сигм.

Далее для данных производился оверсэмплинг со SMOTE. В SMOTE (Способ Передискретизации Синтезированных Меньшинств) создаются элементы в непосредственной близости от уже существующих в меньшем наборе.

В конце производился отбор признаков с помощью алгоритма Boruta. Основная идея алгоритма заключается в сравнении исходных признаков с их теневыми копиями - признаками, полученными с помощью случайного перемешивания значений исходных признаков между строками. Соответственно, признаки, которые мало чем отличаются от теневых будут совершенно не важны для модели.

# **Описание моделей классификации и алгоритмов подбора гиперпараметров**.

## 5.1 Модели классификации

В данной работе используются 4 модели классификации, для которых выбраны гиперпараметры: Метод опорных векторов, k-ближайшие соседи, Градиентный бустинг, Случайный лес. Такой набор моделей был обусловлен тем, что этот список моделей используется авторами в статьях, рассмотренных в разделе 2.

Машины опорных векторов или SVM (машины опорных векторов) — это линейный алгоритм, используемый в задачах классификации и регрессии. Этот алгоритм широко используется на практике и позволяет решать, как линейные, так и нелинейные задачи. Основная задача алгоритма — найти наиболее правильную прямую или гиперплоскость путем разделения данных на два класса [11]. SVM — это алгоритм, который принимает входные данные и возвращает эту разделительную линию.

Ближайшие соседи (k ближайших соседей или KNN) — это метод классификации. Суть метода в следующем: посмотрите на соседей, они тоже будут убегать от вас. Формально метод основан на предположении о компактности: если метрика расстояния между примерами успешно введена, то похожие примеры гораздо чаще находятся в одном классе, чем в разных классах. Для классификации каждого из объектов в тестовой выборке необходимо последовательно выполнить следующие операции: вычисление расстояния до каждого из объектов в обучающей выборке; k выбрать объекты обучающей модели, расстояние до которых минимально; класс объекта, подлежащего классификации, — это тот, который наиболее часто встречается среди k ближайших соседей[12].

Gradient Boosting — это метод машинного обучения для задач классификации и регрессии, который строит прогностическую модель в виде набора слабых прогностических моделей, обычно деревьев решений. Целью любого алгоритма обучения с учителем является определение и минимизация функции потерь. Давайте перейдем к математике улучшения градиента[10]. Функция потерь представляет собой среднеквадратичную ошибку (MSE), для которой приведена формула 1.

, (1)

где i-ое целевое значение;

– i-ое предсказание;

– функция потерь.

Необходимо делать наши прогнозы таким образом, чтобы стандартное отклонение было минимальным. Используя градиентный спуск и обновляя прогнозы на основе скорости обучения, мы ищем значения, при которых MSE минимальна.

Первое предположение линейной регрессии состоит в том, что сумма отклонений = 0, т. е. отклонения должны быть случайным образом распределены вокруг нуля. На рис. 11 показано примерное распределение используемых нами данных.

Теперь давайте подумаем об отклонениях как об ошибках в нашей модели. Хотя древовидные модели не делают такого предположения, если мы подумаем логически (а не статистически) об этом предположении, мы увидим, что мы можем использовать этот шаблон для модели, если мы видим принцип распределения дисперсии. Таким образом, интуиция, стоящая за алгоритмом улучшения градиента, заключается в итеративном применении шаблонов дисперсии и улучшении прогнозов. Как только мы доходим до точки, где дисперсии не показывают закономерностей, мы прекращаем построение нашей модели (в противном случае мы можем переобучиться). Алгоритмически мы минимизируем нашу функцию потерь. Результат: сначала строим простые модели и анализируем ошибки; определить точки, не укладывающиеся в простую модель; Добавьте модели, обрабатывающие сложные случаи, указанные в исходной модели; Собираем все построенные модели и определяем вес каждого предиктора.

Случайный лес — это модель, состоящая из множества деревьев решений. Вместо того, чтобы просто усреднять предсказания разных деревьев, эта модель использует две ключевые концепции, которые делают этот лес случайным. Случайная выборка выборок из набора данных при создании деревьев. При разбиении узлов выбираются случайные наборы параметров. Во время обучения каждое дерево случайного леса обучается на случайной выборке из набора данных. Отбор проб платный. Это позволяет повторно использовать шаблоны в одном и том же дереве. Хотя каждое дерево может широко варьироваться в зависимости от заданного набора обучающих данных, обучающие деревья на разных наборах выборок могут уменьшить общую изменчивость леса без ущерба для точности. При тестировании результат получается путем усреднения прогнозов, полученных от каждого дерева. Вторая основная концепция случайного леса заключается в использовании определенного набора параметров выборки для разделения каждого узла в каждом отдельном дереве. Как правило, размер выборки представляет собой квадратный корень из общего числа параметров. То есть, если каждая запись выборки содержит 16 параметров, в каждом отдельном узле используется 4 параметра. Случайный лес объединяет сотни или тысячи деревьев решений, обучает каждое на отдельной выборке данных и разделяет узлы в каждом дереве с ограниченным набором параметров [9]. Окончательный прогноз делается путем усреднения прогнозов по всем деревьям.

## Алгоритмы подбора гиперпарметров

Рассмотрим алгоритмы, которые будут использоваться для подбора гиперпараметров: grid search, randomized search, байесовская оптимизация, генетический алгоритм. Выбор был сделан, ссылаясь на статьи, рассмотренные в разделе 2, но ранее эти модели не сравнивались между собой. Этот набор позволяет выделить самую эффективную модель для задачи кредитного скоринга.

Наиболее распространенный метод организовать перебор наборов гиперпараметров — выполнить поиск по сетке (Grid Search): для каждого гиперпараметра задается несколько значений; перебираются все комбинации значений разных гиперпараметров, модель обучается и тестируется на каждой из этих комбинаций; выбирается комбинация, при которой модель имеет наилучшее качество[8]. Перечисление некоторых значений гиперпараметров можно производить в логарифмической шкале, так как это позволяет быстро найти правильный порядок параметров при значительном сокращении времени поиска. Блок-схема обучения модели с помощью алгоритма Grid Search показана на рисунке 3.

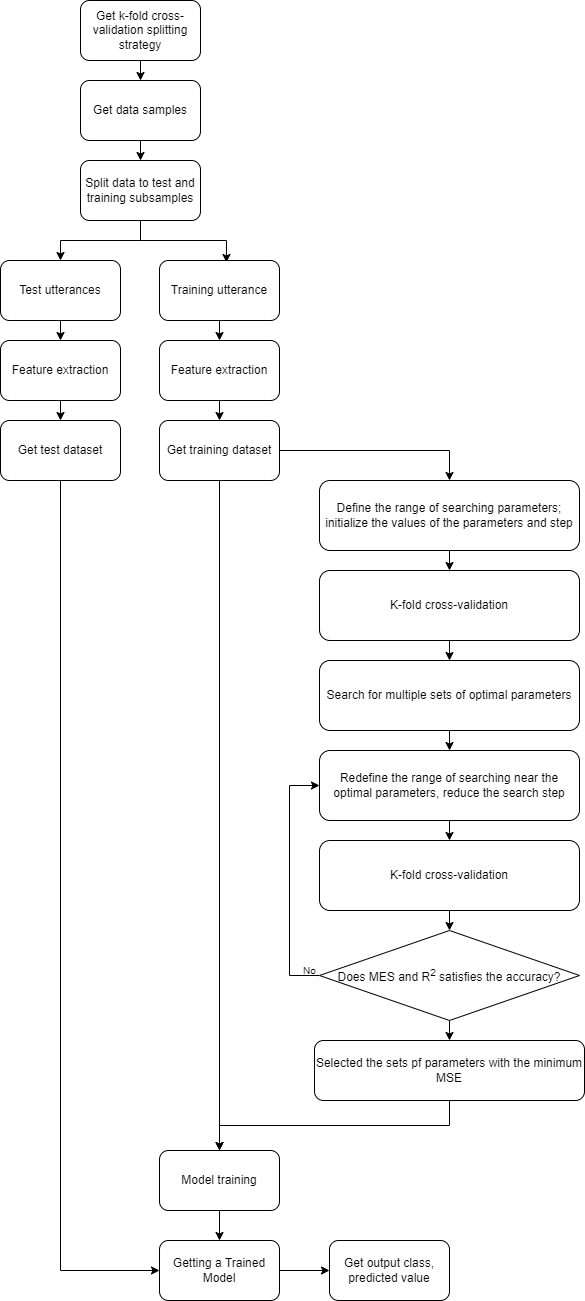


Рисунок 3 – Обучение модели с помощью алгоритма Grid Search

Следующий метод называется рандомизированным поиском (Randomized Search). Для каждого гиперпараметра задается распределение, из которого выбирается его значение, и путем выборки из этих распределений составляется комбинация гиперпараметров. Таким образом, за счет случайного выбора следующей комбинации гиперпараметров можно найти оптимальную комбинацию за меньшее количество итераций. Рисунок 4 хорошо иллюстрирует различия между поиском по сетке и случайным поиском:

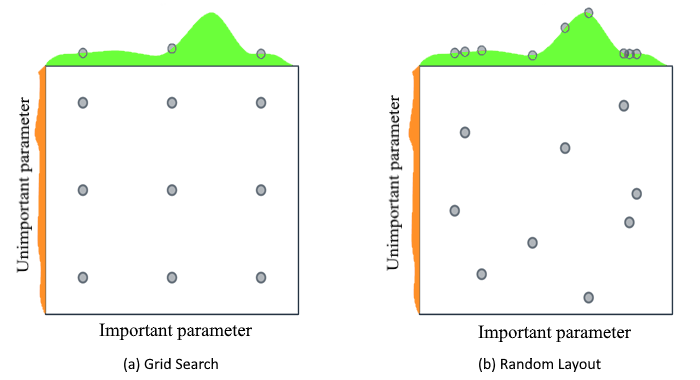


Рисунок 4 – Комбинации гиперпараметров (а – поиск по сетке, b – случайный поиск)

Идея, представленная на этой картинке, заключается в следующем. Качество нашей модели как функция гиперпараметров является функцией многих переменных с нетривиальной поверхностью. Но эта поверхность может гораздо меньше зависеть от одной из своих переменных, чем от другой.

Существует много способов исследовать пространство поиска более эффективно, чем случайным образом. Их основная идея проста: если какая-то область пространства оказывается хорошей, ее нужно исследовать дальше. Такие методы позаботятся о процессе «масштабирования» и приводят к значительно лучшим решениям за значительно короткое время. Один из таких методов называется байесовской оптимизацией. Байесовская оптимизация — итерационный метод, позволяющий оценить оптимум функции без ее дифференцирования [13]. Кроме того, на каждой итерации метод указывает, в какой следующей точке мы, скорее всего, улучшим нашу текущую оценку оптимума. Это позволяет значительно сократить количество вычислений функций, каждое из которых может занимать достаточно много времени. Выбор гиперпараметров также можно сформулировать как задачу, которую можно решить с помощью байесовской оптимизации. Например, пусть нашей функцией будет значение метрик проверки в зависимости от текущей комбинации гиперпараметров. Его вычисление требует много времени (модель должна быть обучена и проверена), и мы не можем вычислить градиенты этой функции по ее переменным (нашим гиперпараметрам).

Генетический алгоритм - это метаэвристика, вдохновленная естественным отбором; они используются в задачах оптимизации и поиска в целом и обычно основаны на наборе функций, таких как мутация, кроссовер и отбор [20].

Алгоритм действует следующим способом:

1. Генерирует случайно выбранную совокупность (разные наборы гиперпараметров); это поколение 0;
2. Оценивает значение пригодности каждого человека в популяции с точки зрения машинного обучения, получает баллы перекрестной проверки;
3. Генерируйет новое поколение, используя несколько генетических операторов;
4. Повторяет шаги 2 и 3, пока не достигнет критерия остановки.

В данном исследовании используется классический генетический алгоритм, без модификаций. Генерируется 10 популяций с 6 особами в каждой. Для отбора будет использоваться турнирный метод с размером 3. Из каждой популяции будут случайно отбираться три индивида, среди который наиболее приспособленный будет родителем следующего поколения. Этот процесс будет повторяться до тех пор, пока число родителей не будет равно размеру начальной популяции.

# 5. Сравнение результатов.

Несмотря на популярность машинного обучения, во многих её сферах до сих пор не сформировалась единая теоретическая концепция. Исключением не стала и рассматриваемая область. Хоть и существуют некоторые общие рекомендации к применению метрик для некоторых задач, конечное решение лежит на плечах аналитика. Для сравнения алгоритмов для подбора гиперпарметров были взяты следующие метрики: F1, Accuracy, ROC-AUC, Recall, Precision. Они были взяты, т.к. они являются наиболее популярными и чаще всего используются сообществом для сравнения алгоритмов. Данные метрики будут использоваться для сравнения в таблицах 1-4.

Сравним результаты для метода опорных векторов. Для данного алгоритма перебирались параметры «C» и «kernel». C – параметр регуляризации, представляет собой штраф l2. Kernel – тип ядра, который используется в алгоритме. Возможные значения параметра "C" - 0.1, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, для значения "kernel" - "linear", "poly", "rbf", "sigmoid". Результаты представлены в таблице 1.

Таблица 1 – Результаты обучения модели метода опорных векторов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Метод обучения \ метрики | Время обучения | F1 | Accuracy | ROC-AUC | Recall | Precision | Параметры модели |
| Grid search | 868 сек. | 0,6715 | 0,6775 | 0,7499 | 0,6737 | 0,6693 | C=5,  Kernel=’rbf’ |
| Randomized Search | 231 сек. | 0,6713 | 0,6776 | 0,7500 | 0,6726 | 0,6699 | C=4,  Kernel=’rbf’ |
| Bayes Search | 1352 сек. | 0,6715 | 0,6775 | 0,7499 | 0,6737 | 0,6693 | C=5,  Kernel=’rbf’ |
| Genetic Algorithm | 4303 сек. | 0,6715 | 0,6775 | 0,7499 | 0,6737 | 0,6693 | C=5,  Kernel=’rbf’ |

Быстрее всего справился Randomized Search, показав похожие результаты, по сравнению с другими методами подбора гиперпараметров.

Сравним результаты для k-ближайших соседей. Для данного алгоритма перебирались параметры «n\_neighbors», «weights» и «algorithm». n\_neighbors – число соседей. weights – функция для взвешивания соседей. algorithm – алгоритм, используемый для вычисления ближайших соседей. Возможные значения: все соседи с одинаковым весом или пропорционально расстоянию до соседа. Возможные значения параметра "n\_neighbors" – диапазон значений от 1 до 30 включительно, для значения "weights" - "uniform", "distance", для значения "algorithm" – "auto", "ball\_tree", "kd\_tree", "brute". Результаты для данного алгоритма представлены в таблице 2.

Таблица 2 - Результаты обучения модели k-ближайших соседей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Метод обучения \ метрики | Время обучения | F1 | Accuracy | ROC-AUC | Recall | Precision | Параметры модели |
| Grid search | 72 сек. | 0,6900 | 0,6776 | 0,7462 | 0,7333 | 0,6516 | N\_neightbors=27,  Weights=’distance’,  Algorithm=’auto’ |
| Randomized Search | 4 сек. | 0,6900 | 0,6776 | 0,7462 | 0,7333 | 0,6516 | N\_neightbors=27,  Weights=’distance’,  Algorithm=’brute’ |
| Bayes Search | 80 сек. | 0,6900 | 0,6776 | 0,7462 | 0,7333 | 0,6516 | N\_neightbors=27,  Weights=’distance’,  Algorithm=’kd\_tree’ |
| Genetic Algorithm | 186 сек. | 0,6900 | 0,6776 | 0,7462 | 0,7333 | 0,6516 | N\_neightbors=27,  Weights=’distance’,  Algorithm=’kd\_tree’ |

Быстрее всего справился Randomized Search, показав похожие результаты, по сравнению с другими методами подбора гиперпараметров.

Сравним результаты, указанные на таблице 3, для градиентного бустинга. Для данного алгоритма перебирались параметры «max\_depth», «n\_estimators», «criterion». max\_depth – максимальная глубина дерева. n\_estimators – число деревьев. criterion – функция ошибки. Возможные значения параметра "max\_depth" - 2, 3, 4, 5, 6, 7, 8, 9, 10, для значения "n\_estimators" - 10, 25, 50, 100, 500, 1000, 2000, 3000, 400, 5000, для значения "criterion" - "friedman\_mse", "squared\_error".

Таблица 3 - Результаты обучения модели градиентного бустинга

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Метод обучения \ метрики | Время обучения | F1 | Accuracy | ROC-AUC | Recall | Precision | Параметры модели |
| Grid search | 5909 сек. | 0,7199 | 0,7367 | 0,8162 | 0,6915 | 0,7507 | Criterion=’squared\_error’, n\_estimators=500,  Max\_depth=2 |
| Randomized Search | 450 сек. | 0,7198 | 0,7365 | 0,8127 | 0,6919 | 0,7502 | Criterion=’squared\_error’, n\_estimators=100,  Max\_depth=4 |
| Bayes Search | 1884 сек. | 0,7198 | 0,7365 | 0,8127 | 0,6919 | 0,7502 | Criterion=’squared\_error’, n\_estimators=100,  Max\_depth=4 |
| Genetic Algorithm | 4826 сек. | 0,7194 | 0,7351 | 0,8120 | 0,6940 | 0,7467 | Criterion=’friedman\_mse’, n\_estimators=100,  Max\_depth=5 |

Быстрее всего справился Randomized Search, показав похожие результаты, по сравнению с другими методами подбора гиперпараметров.

Сравним результаты, которые расположены в таблице 4, для случайного леса. Для данного алгоритма перебирались параметры «max\_depth», «n\_estimators», «criterion», «bootstrap». max\_depth – максимальная глубина дерева. n\_estimators – число деревьев. criterion – критерий по разбиению признака, по критерию Джини, энтропии или кросс-энтропия (логистическая функция потерь). bootstrap - спользуются ли образцы начальной загрузки при построении деревьев. Возможные значения параметра "max\_depth" - 2, 3, 4, 5, 6, 7, 8, 9, 10, для значения "n\_estimators" - 10, 25, 50, 100, 500, 1000, 2000, 3000, 4000, 5000, для значения "criterion" -"gini", "entropy", "log\_loss", для значения "bootstrap" – True, False.

Таблица 4 - Результаты обучения модели случайного леса

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Метод обучения \ метрики | Время обучения | F1 | Accuracy | ROC-AUC | Recall | Precision | Параметры модели |
| Grid search | 5691 сек. | 0,7165 | 0,7251 | 0,7995 | 0,7100 | 0,7232 | Criterion=’entrory’,  N\_estimators=5000,  Max\_depth=10,  Bootstrap=True |
| Randomized Search | 154 сек. | 0,7161 | 0,7244 | 0,7988 | 0,7104 | 0,7219 | Criterion=’gini’,  N\_estimators=2000,  Max\_depth=10,  Bootstrap=True |
| Bayes Search | 1099 сек. | 0,7165 | 0,7251 | 0,7995 | 0,7100 | 0,7273 | Criterion=’entrory’,  N\_estimators=5000,  Max\_depth=10,  Bootstrap=True |
| Genetic Algorithm | 4577 сек. | 0,7165 | 0,7251 | 0,7995 | 0,7100 | 0,7273 | Criterion=’entrory’,  N\_estimators=5000,  Max\_depth=10,  Bootstrap=True |

Быстрее всего справился Randomized Search, показав похожие результаты, по сравнению с другими методами подбора гиперпараметров.

На рисунке 5 указаны ROC-AUC графики всех моделей по всем алгоритмам.

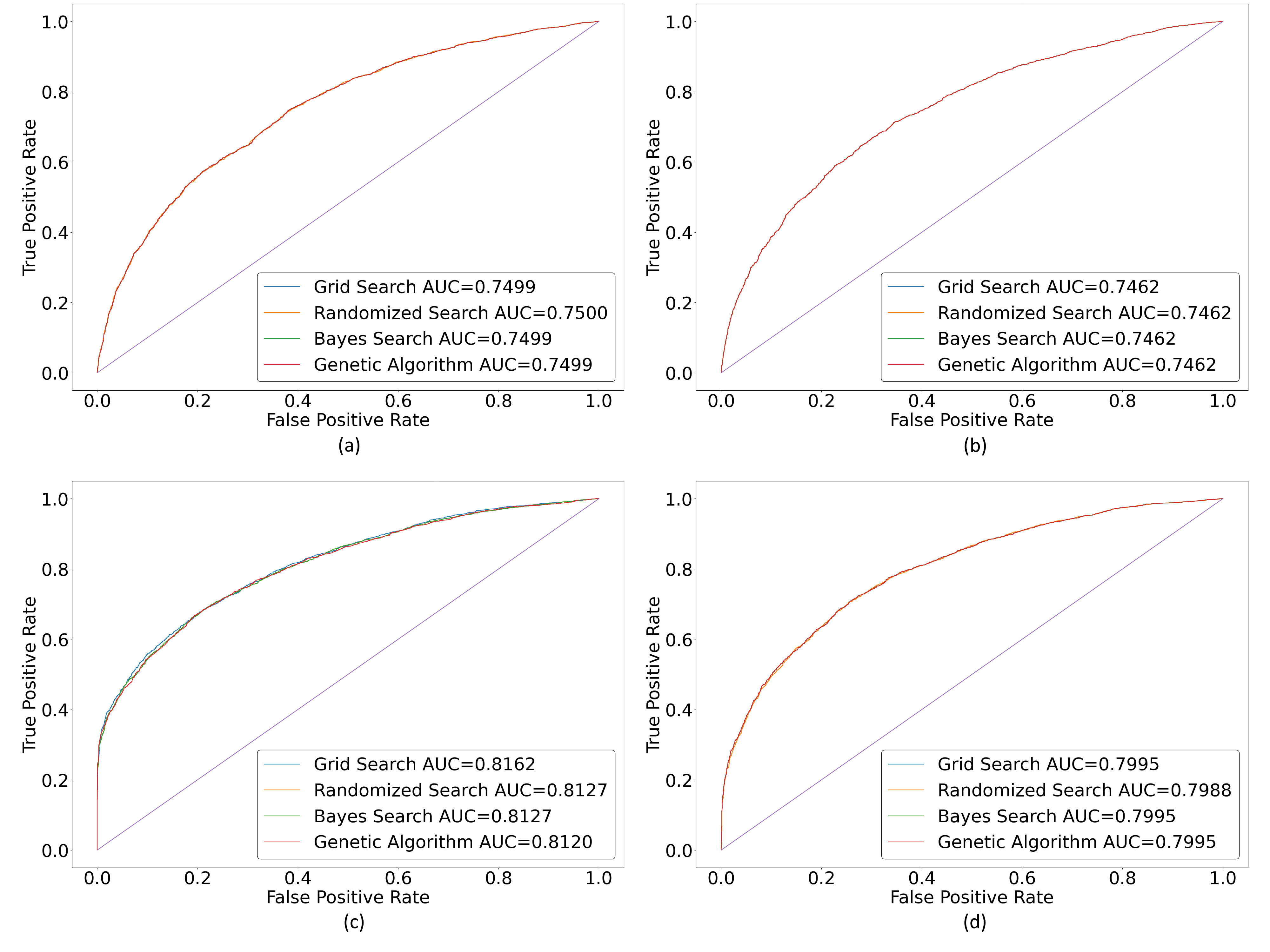


Рисунок 5 – ROC-AUC графики обученных моделей классификации (a – SVC, b – K-Nearest Neighbors, c – Gradient Boosting, d – Random Forest)

## 6. Выводы

Цель этой статьи состояла в том, чтобы сравнить четыре различных алгоритма поиска лучших гиперпараметров для нейронной сети. Используемые алгоритмы: Grid Search, Random Search, Баесовская Оптимизация, Генетический алгоритм. Набор данных [SF-DST] Credit Scoring был выбран для проведения исследования. Процесс оценки был сосредоточен на Время обучения, F1, Accuracy, ROC-AUC и Recall Precision.

В результате данной работы было выявлено, что все методы при исходном наборе данных выдают одинаковый набор гиперпараметров для модели при различном времени работы алгоритма. В итоге, использование более вычислительно сложных алгоритмов из данного набора, которые затрачивают больше времени, не является лучшим решением и достаточно использовать менее сложные для получения подобных итоговых значений. Резюмируя выше сказанное, мы делаем выводы, что из 4 алгоритмов лучше будет использовать Randomized Search, который затрачивает меньше всего времени, тем самым увеличивая производительность.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] D. Marinov, D. Karapetyan, Hyperparameter Optimisation with Early Termination of Poor Performers, 2019.

[2] J. Bergstra, Y. Bengio, Random Search for Hyper-Parameter Optimization, Journal of Machine Learning Research 13, pp. 281-305, 2012.

[3] Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. J. Glob. Optim. 1998, 13, 455–492.

[4] Li L, Jamieson K, DeSalvo G, et al. Hyperband: A novel bandit-based approach to hyperparameter optimization[J]. The Journal of Machine Learning Research, 2017, 18(1): 6765-6816.

[5] Chiara Di Francescomarinoa, Marlon Dumas, Marco Federici, Chiara Ghidini, Fabrizio Maria Maggi, Williams Rizzi, Luca Simonettoa. Genetic algorithms for hyperparameter optimization in predictive business process monitoring, 2018, 74(1): 67-83.

[6] [SF-DST] Credit Scoring // Kaggle URL: https://www.kaggle.com/c/sf-dst-scoring (дата обращения: 06.12.20222).

[7] Hutter F, Lücke J, Schmidt-Thieme L. Beyond manual tuning of hyperparameters[J]. KIKünstliche Intelligenz, 2015, 29(4): 329-337.

[8] Учебник по машинному обучению // Яндекс Академия URL: https://academy.yandex.ru/handbook/ml/article/podbor-giperparametrov (дата обращения: 06.12.20222).

[9] Реализация и разбор алгоритма «случайный лес» на Python // Типичный программист URL: https://tproger.ru/translations/python-random-forest-implementation/ (дата обращения: 06.12.20222).

[10] Градиентый бустинг — просто о сложном // Neurohive URL: https://neurohive.io/ru/osnovy-data-science/gradientyj-busting/ (дата обращения: 06.12.20222).

[11] Краткий обзор алгоритма машинного обучения Метод Опорных Векторов (SVM) // Habr URL: https://habr.com/ru/post/428503/ (дата обращения: 06.12.20222).

[12] Метод k-ближайших соседей (k-nearest neighbour) // Proglib URL: https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19 (дата обращения: 06.12.20222).

[13] The Beauty of Bayesian Optimization, Explained in Simple Terms // Towards data science URL: https://towardsdatascience.com/the-beauty-of-bayesian-optimization-explained-in-simple-terms-81f3ee13b10f (дата обращения: 06.12.20222).

[14] Hussain Alibrahim, Simone A. Ludwig, Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization, 2021

[15] Automatic Machine Learning Method for Hyperparameter Search, Journal of Physics: Conference Series, Minglan Su et al 2021 J. Phys.: Conf. Ser. 1802 032082

[16] Enhanced machine learning tree classifiers for lithology identification using Bayesian optimization, Solomon Asante-Okyere, Chuanbo Shen, Harrison Osei, 29 June 2022

[17] Evaluating ML-based DDoS Detection with Grid Search Hyperparameter Optimization, Odnan Ref Sanchez, Matteo Repetto, Alessandro Carrega, Raffaele Bolla, pp402 – 408

[18] Hyperparameters search methods for machine learning linear workflows, Klara Peskova, Roman Neruda, pp. 1205-1210 2019, 18th IEEE International Conference on Machine Learning and Applications (ICMLA)

[19] Hyperparameter Optimization on Support Vector Machine using Grid Search for Classifying Thalassemia Data, Afifah Rofi Laeli, Zuherman Rustam, Sri Hartini, Faisa Maulidina, Jane Eva Aurelia 2020 International Conference on Decision Aid Sciences and Application (DASA) | 978-1-7281-9677-0/20

[20] Вы все еще используете поиск по сетке для оптимизации гиперпараметров? // Scmax URL: https://scmax.ru/articles/41660/ (дата обращения: 17.12.2022).

[21] Törn, A.; Žilinskas, A. Global Optimization; Springer: Berlin, Germany, 1989.

[22] Mongeau, M.; Karsenty, H.; Rouzé, V.; Hiriart-Urruty, J.B. Comparison of public-domain software for black-box global optimization. Optim. Methods Softw. 1998, 13, 203–226.

[23] Liberti, L.; Maculan, N. Global Optimization: From Theory to Implementation; Springer Optimization and Its Applications; Springer: Berlin, Germany, 2006.

[24] Zhigljavsky, A.; Žilinskas, A. Stochastic Global Optimization; Springer Optimization and Its Applications; Springer: Berlin, Germany, 2007

[25] Calandra, R.; Peters, J.; Rasmussen, C.E.; Deisenroth, M.P. Manifold Gaussian processes for regression. In Proceedings of the 2016 International Joint Conference on Neural Networks, Vancouver, BC, Canada, 24–29 July 2016; pp. 3338–3345.

[26] Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. In Proceeding of the Advances in Neural Information Processing Systems 25, Neural Information Processing Systems(NIPS), Lake Tahoe, NV, USA, 3–8 December 2012; pp. 2951–2959