|  |  |
| --- | --- |
|  | **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 \_\_\_\_\_\_«Международных образовательных программ»\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_«Программное обеспечение ЭВМ и информационные технологии»\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***Применение алгоритма случайного леса для машинного обучения при исключении добросовестных соискателей кредита***

Студент \_\_\_ИУ7И-86Б\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Нгуен Т.Т.\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_Романова Т.Н.\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(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 ИУ7

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Рудаков И.В.\_

(И.О.Фамилия)

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы бакалавра**

Студент группы \_\_ИУ7И-86Б\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Нгуен Тхань Тхиен\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Тема квалификационной работы Применение алгоритма случайного леса для машинного обучения при исключении добросовестных соискателей кредита.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (НИР кафедры, заказ организаций и т.п.)

\_НИР кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема квалификационной работы утверждена распоряжением по факультету \_ИУ\_\_\_\_\_\_\_\_\_\_\_\_\_\_ № \_03.02.01-04.03/14\_от « 11 » декабря 2021 г.

**Часть 1.** Аналитический раздел***\_\_\_\_\_\_\_\_\_\_***

Изучение работу дерева решений и изучение алгоритма случайного леса. Анализ преимуществ и недостатков алгоритма случайного леса. Анализ существующих данных и методов для подготовки данных.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Часть 2.*** Конструкторский раздел\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Схемы реализации, описания логики конкретных процессов при построении и тестировании алгоритмов случайного леса.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Часть 3.*** Технологический раздел\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Осуществить выбор языка и среды программирования. Изучение вспомогательные библиотеки для обработки и предварительной обработки данных, чтобы подготовиться к обучению моделей машинного обучения. Реализовать пользовательский интерфейс для ввода данных и отображения прогнозов.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Часть 4.*** Исследовательский раздел\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Провести анализ и оценить эффективность классификации модели. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление квалификационной работы:***

Расчетно-пояснительная записка на \_50-70\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « » \_\_\_\_\_\_\_ 2022 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до « \_\_ » \_\_\_\_\_\_\_\_\_ 2022 г.

**Руководитель квалификационной работы** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Романова Т.Н.\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_**Нгуен Т.Т.\_\_**\_\_\_**

(Подпись, дата) (И.О.Фамилия)

**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** **\_**ИУ**\_\_\_\_\_\_\_\_** УТВЕРЖДАЮ

**КАФЕДРА** **\_**ИУ7**\_\_\_\_\_\_\_\_\_\_** Заведующий кафедрой \_\_\_\_ИУ 7\_\_

(Индекс)

**ГРУППА** **\_**ИУ7И-86Б**\_\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Рудаков И.В.\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**КАЛЕНДАРНЫЙ ПЛАН**

**выполнения выпускной квалификационной работы**

студента: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Нгуен Тхань Тхиен\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

Тема квалификационной работы Применение алгоритма случайного леса для машинного обучения при исключении добросовестных соискателей кредита.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Наименование этапов выпускной квалификационной работы** | **Сроки выполнения этапов** | | **Отметка о выполнении** | |
| **план** | **факт** | **Должность** | **ФИО, подпись** |
|  | Задание на выполнение работы. Формулирование проблемы, цели и задач работы | *24.10.2021*  *Планируемая дата* |  | Руководитель ВКР |  |
|  | 1 часть Аналитический раздел | *24.12.2021*  *Планируемая дата* |  | Руководитель ВКР |  |
|  | Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач | *24.12.2021*  *Планируемая дата* |  | Заведующий кафедрой |  |
|  | 2 часть Конструкторский раздел | *06.04.2022*  *Планируемая дата* |  | Руководитель ВКР |  |
|  | 3,4 части Технологический и исследовательский разделы | *18.05.2022*  *Планируемая дата* |  | Руководитель ВКР |  |
|  | 1-я редакция работы | *20.05.2022*  *Планируемая дата* |  | Руководитель ВКР |  |
|  | Подготовка доклада и презентации | *20.05.2022*  *Планируемая дата* |  |  |  |
|  | Заключение руководителя | *24.05.2022*  *Планируемая дата* |  | Руководитель ВКР |  |
|  | Допуск работы к защите на ГЭК (нормоконтроль) | *27.05.2022*  *Планируемая дата* |  | Нормоконтролер |  |
|  | Внешняя рецензия | *24.05.2022*  *Планируемая дата* |  |  |  |
|  | Защита работы на ГЭК | *06.06.2022*  *Планируемая дата* |  |  |  |

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

# РЕФЕРАТ

Расчетно-пояснительная записка 60 страницы, 28 рисунков, 2 таблицы, 16 листингов и 7 источников.

Работа посвящена реализации алгоритма случайного леса для машинного обучения с целью исключения недобросовестных соискателей в процессе одобрения кредита.

В работе рассмотрены реализации алгоритм случайного леса, сравнение эффективности конкретных реализаций. Оценить эффективность алгоритма. Создание приложений, помогающих пользователям прогнозировать конкретные случаи клиентов.

**СОДЕРЖАНИЕ**

[РЕФЕРАТ 5](#_Toc105116226)

[Введение 8](#_Toc105116227)

[1. Аналитический раздел 9](#_Toc105116228)

[1.1. Обзор алгоритма случайного леса 9](#_Toc105116229)

[1.2. Понятие дерево решений 9](#_Toc105116230)

[1.2.1. Анализ построения дерева решений 10](#_Toc105116231)

[1.2.2. Построение дерева решений 11](#_Toc105116232)

[1.2.3. Критерии расщепления 12](#_Toc105116233)

[1.2.4. Критерии останова 14](#_Toc105116234)

[1.2.5. Методы стрижки дерева 15](#_Toc105116235)

[1.2.6. Обработка пропущенных значений 15](#_Toc105116236)

[1.2.7. Методы построения деревьев 16](#_Toc105116237)

[1.3. Алгоритм случайного леса 17](#_Toc105116238)

[1.3.1. Out–Of–Bag 19](#_Toc105116239)

[1.4. Данные для построения и тестирования деревьев решений для классификации добросовестных соискателей кредита 20](#_Toc105116240)

[1.4.1. Структура данных 21](#_Toc105116241)

[1.4.2. Анализ отношений между переменными и метками 22](#_Toc105116242)

[1.4.2.1. Числовые переменные 22](#_Toc105116243)

[1.4.2.2. Категориальные переменные 25](#_Toc105116244)

[1.4.3. Дисбаланс распределения категорий выборки 27](#_Toc105116245)

[1.4.3.1. Избыточная выборка 27](#_Toc105116246)

[1.4.3.2. Недостаточная выборка 28](#_Toc105116247)

[1.5. Оценить эффективность модели машинного обучения 29](#_Toc105116248)

[1.5.1. Матрица путаницы – confusion matrix 29](#_Toc105116249)

[1.5.2. Точность - accuracy 30](#_Toc105116250)

[Вывод 31](#_Toc105116251)

[2. Конструкторский раздел 32](#_Toc105116252)

[2.1. Описание алгоритма дерева решений 32](#_Toc105116253)

[2.1.1. Схема алгоритма 32](#_Toc105116254)

[2.1.2. Описание алгоритма 33](#_Toc105116255)

[2.2. Описание алгоритма случайного леса 34](#_Toc105116256)

[2.2.1. Схема алгоритма 34](#_Toc105116257)

[2.2.2. Описание алгоритма 35](#_Toc105116258)

[2.3. Приложение для прогнозирования рисков 35](#_Toc105116259)

[Вывод 37](#_Toc105116260)

[3. Технологический раздел 38](#_Toc105116261)

[3.1. Язык программирования 38](#_Toc105116262)

[3.2. Редактор кода 38](#_Toc105116263)

[3.3. Подготовка набора данных 38](#_Toc105116264)

[3.3.1. Загрузить данные в программу 38](#_Toc105116265)

[3.3.2. Обработка дисбаланса данных 38](#_Toc105116266)

[3.3.3. Смешивание и разделение данных 39](#_Toc105116267)

[3.4. Вычислить энтропии и прирост информации 39](#_Toc105116268)

[3.5. Построение дерева решений 40](#_Toc105116269)

[3.6. Случайный лес класса 41](#_Toc105116270)

[3.7. Сохранение и загрузка случайной леса модели 42](#_Toc105116271)

[3.8. Оценка эффективности обученной модели 42](#_Toc105116272)

[3.9. Пользовательский интерфейс 43](#_Toc105116273)

[4. Исследовательский раздел 45](#_Toc105116274)

[4.1. Описание экспериментов 45](#_Toc105116275)

[4.2. Оценка эффективности модели 45](#_Toc105116276)

[4.2.1. Точность 45](#_Toc105116277)

[4.2.2. Матрица путаницы 46](#_Toc105116278)

[4.3. Приложение для прогнозирования риска соискателей в процессе одобрения кредита 47](#_Toc105116279)

[4.3.1. Прогнозирования соискатели с высоким риском 47](#_Toc105116280)

[4.3.2. Прогнозирования соискатели с низким риском 48](#_Toc105116281)

[ЗАКЛЮЧЕНИЕ 50](#_Toc105116282)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 51](#_Toc105116283)

[ПРИЛОЖЕНИЕ А 52](#_Toc105116284)

[ПРИЛОЖЕНИЕ Б 56](#_Toc105116285)

[ПРИЛОЖЕНИЕ В 58](#_Toc105116286)

[ПРИЛОЖЕНИЕ Г 60](#_Toc105116287)

[ПРИЛОЖЕНИЕ Д 60](#_Toc105116288)

# Введение

В современном мире, мы наблюдаем взрыв интереса к машинным обучениям, которые успешно применяются в самых различных областях, как бизнесе, медицине, технике,... Невозможно не признать огромную пользу, которую приносит машинное обучение.

В данном работе, мы будем изучать алгоритм случайного леса. Случайный лес - один из самых потрясающих алгоритмов машинного обучения, придуманные Лео Брейманом и Адель Катлер ещё в прошлом веке. Алгоритм строится на основе алгоритма дерева решений, то есть случайный лес состоит из множества деревьев решений, каждое дерево в лесу строится путем случайного выбора выборки данных. Это также происхождение имени “**случайный лес**”.

**Цель** **работы**: Исследование и анализирование алгоритма машинного обучения случайного леса с целью исключения недобросовестных соискателей в процессе одобрения кредита. Требуется решить следующие задачи:

* Анализ алгоритма случайного леса.
* Выбор и построение дерева решений.
* Критерии разделения.
* Критерий остановки.
* Выбор данных.
* Подготовка данных.
* Построение модели случайного леса.
* Оценка эффективности обученной модели случайного леса.
* Создавать приложение, для прогнозирования риска соискателей в процессе одобрения кредита.

1. **Аналитический раздел**
   1. **Обзор алгоритма случайного леса**

Алгоритм случайного леса основан на контролируемом обучении и его можно использовать как для задач регрессии, так и для задач классификации.

Случайный лес можно рассматривать как совокупность алгоритмов множественных деревьев решений со случайной выборкой. Этот алгоритм предназначен для устранения недостатков алгоритма дерева решений. Так что для понять работу случайного леса, **сначала нужно понять работу дерева решений**, так как случайный лес основан на деревьях решений.

* 1. **Понятие дерево решений**

Деревья решений можно использовать для решения задач классификации и регрессии, но в данной работе, анализирование только задачи классификации.

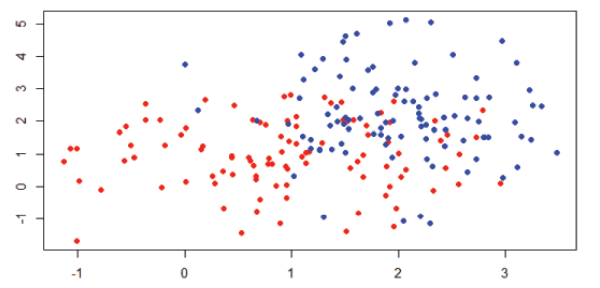
Рассмотрения размеченную выборку объектов {(xi, yi)}Ni=1, где xi ∈ R2 - признаковое описание объекта в двумерном пространстве, а yi ∈ {0, 1} - метка класса.

Пример:

Дано точечное множество имеет координаты (x, y), все точки на относятся к 2 классам: красный и синий. Построение дерева решений, чтобы классифицировать эти точки.

|  |  |  |  |
| --- | --- | --- | --- |
| xi | | yi | |
| x(Координаты) | y(Координаты) | 0(красный) | 1(Синий) |

*Таб.1 -* Точечное множество*.*



*Рис.1 - Исходные данные задачи классификации.*

* + 1. **Анализ построения дерева решений**

Структура дерева решений, состоящая из:

* Узлов-листьев (решений), каждый из которых представляет определенный класс;
* Узлов проверки (принятия решений), содержащих тестовые процедуры, которые должны быть выполнены по отношению к одному из атрибутов.
* Из узла проверки выходят ветви (случаи ответа), количество которых соответствует количеству возможных исходов тестовой процедуры.

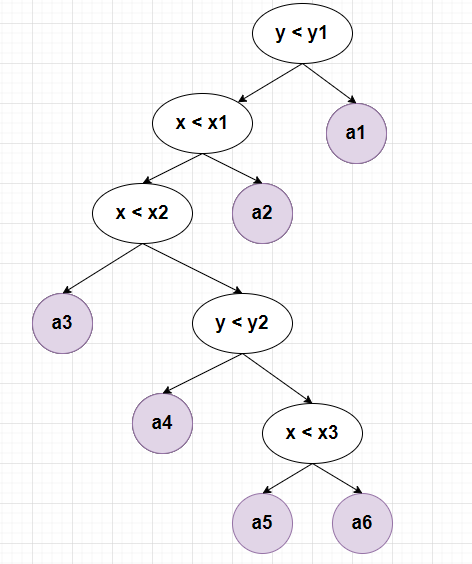
Легко убедиться, что для любой выборки можно построить решающее дерево, не допускающее на ней ни одной ошибки - даже с простыми одномерными предикатами можно сформировать дерево, в каждом листе которого находится ровно по одному объекту выборки. Скорее всего, это дерево будет переобученным и не сможет показать хорошее качество на новых данных. Можно было бы поставить задачу поиска дерева, которое является минимальным (с точки зрения количества листьев) среди всех деревьев, не допускающих ошибок на обучении - в этом случае можно было бы надеяться на наличие у дерева обобщающей способности. К сожалению, эта задача до сих пор не решен, и поэтому приходится ограничиваться жадными алгоритмами построения дерева.

Таким образом, конкретный метод построения решающего дерева определяется:

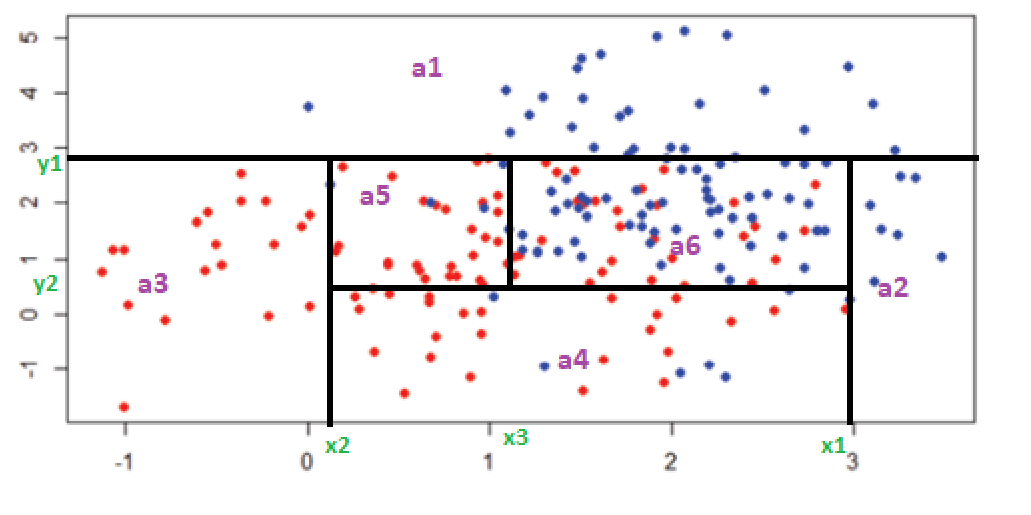
* Критерии расщепления.
* Критерием останова.
* Методом стрижки.
  + 1. **Построение дерева решений**

При каждом делении все объекты делятся на две более мелкие группы, т.е. расматриваемая в каждом из узлов задача разбивается на две более мелкие подзадачи. Заданием максимального числа объектов в вершине-листе дерева устанавливается один из возможных критериев останова для алгоритма.

Полученное в итоге работы алгоритма дерево можно изобразить несколькими способами, как показано на рис.2. Таким образом, можно достаточно качественно классифицировать рассматриваемую выборку объектов при помощи всего одного дерева решений, если в качестве ответа для тестового объекта, попавшего в ячейку Ai , выдавать номер наиболее часто встречающегося в этой ячейке класса.



*Рис.2 - Один вариант изображения решающих деревьев.*



*Рис.3 - Визуальное дерево решений на графике.*

Шаги для построения дерева решений по приведенному выше варианту (рис.3):

* Проводим линию y = y1, линия делит на 2 части, в которых: часть a1 включает только синие точки.
* В другой половине, проводим линию x = x1, линия делит на 2 части, в которых: часть a2 включает только синие точки.
* В другой половине, проводим линию x = x2, линия делит на 2 части, в которых: часть a3 включает только красные точки.
* В другой половине, проводим линию y = y2, линия делит на 2 части, в которых: часть a4 включает большинство красные точки.
* В другой половине, проводим линию x = x3, линия делит на 2 части, в которых: часть a5 включает большинство красные точки.

На самом деле, в более сложных примерах, построение дерева решений сильно зависит от критериев разделения и критерии останова в каждом узле.

* + 1. **Критерии расщепления**

Критерий расщепления - такой атрибут, при котором объекты подмножеств, получаемых в результате разбиения, являлись бы представителями одного класса или были максимально приближены к такому разбиению.

На рис.1 мы увидим что в середине объекты разных классов сильно перемешаны, при помощи дерева решений с такой выборкой достаточно удобно работать: на каждом шаге необходимо выбирать признак и значения порога, по которому происходит оптимальное по заданному критерию разбиение. При решении прикладных задач часто используются следующие критерии:

* **Примесь gini:**

Где pj — вероятность класса j.

**Примесь gini** измеряет частоту, с которой любой элемент набора данных будет неправильно помечен, когда он будет помечен случайным образом.

Минимальное значение индекса **gini** равно 0. Это происходит, когда узел чистый, это означает, что все содержащиеся в узле элементы относятся к одному уникальному классу. Поэтому этот узел больше не будет разделен. Таким образом, оптимальное разделение выбирается по признакам с меньшим индексом **gini**. Более того, он получает максимальное значение, когда вероятность двух классов одинакова.

Ginimin = 1 – 12 = 0

Ginimax = 1 – (0.52 + 0.52) = 0.5

* **Entropy:**

Где pj - вероятность класса j.

Entropy - это мера информации, указывающая на неупорядоченность признаков с целью. Подобно индексу **gini**, оптимальное разделение выбирается по признаку с меньшей энтропией. Он получает максимальное значение, когда вероятность двух классов одинакова, а узел является чистым, когда entropy имеет минимальное значение, равное 0.

Entropymin = = 0

Entropymax = = 1

Энтропия - это мера беспорядка или неопределенности, а цель моделей машинного обучения и специалистов по Data Science в целом состоит в том, чтобы эту неопределенность уменьшить.

В вычислительном отношении **entropy** более сложна, поскольку использует логарифмы, и, следовательно, вычисление индекса **gini** будет происходить быстрее.

* **Information Gain:**

Прирост информации атрибута A в S определяется:

Gain(S, A) = Entropy(S) -

Где Values(A) это множество всех значений A, и Sv = {x | x in S, xa = v}.

Второй часть Gain(S, A) измеряет оставшуюся информацию, когда s делится на подмножества в соответствии со значениями A.

Из выше формулы следует, что чем меньше (то есть чем более однородны элементы), тем больше Gain(S, A). Таким образом, чем больше Gain(S, A), тем более однородны элементы.

* + 1. **Критерии останова**

Можно придумать большое количестве критериев останова. Перечислим некоторые ограничения и критерии:

* Ограничение максимальной глубины дерева
* Ограничение минимального числа объектов в листе
* Ограничение максимального количества листьев в дереве
* Останов в случае, если все объекты в листе относятся к одному классу

С помощью грамотного выбора подобных критериев и их параметров можно существенно повлиять на качество дерева. Тем не менее, такой подбор является трудозатратным[1].

* + 1. **Методы стрижки дерева**

Стрижка дерева является альтернативой критериям останова, описанным выше. При использовании стрижки сначала строится переобученное дерево (например, до тех пор, пока в каждом листе не окажется по одному объекту), а затем производится оптимизация его структуры с целью улучшения обобщающей способности. Существует ряд исследований, показывающих, что стрижка позволяет достичь лучшего качества по сравнению с ранним остановом построения дерева на основе различных критериев.

Тем не менее, на данный момент методы стрижки редко используются и не реализованы в большинстве библиотек для анализа данных. Причина заключается в том, что деревья сами по себе являются слабыми алгоритмами и не представляют большого интереса, а при использовании в композициях они либо должны быть переобучены (в случайных лесах), либо должны иметь очень небольшую глубину (в бустинге), из-за чего необходимость в стрижке отпадает.

Поэтому на данном работе, не использование стрижки дерева методы**.**

* + 1. **Обработка пропущенных значений**

Одним из основных преимуществ решающих деревьев является возможность работы с пропущенными значениями. Рассмотрим некоторые варианты.

* Построение суррогатных предикатов в каждой вершине. Так называется предикат, который использует другой признак, но при этом дает разбиение, максимально близкое к данному.
* Использование минимальное, максимальное, среднее или наиболее распространенные значения атрибута вместо потерянных значений.

Предположим, что D' представляет подмножество выборок в D, у которых нет пропущенных значений атрибута a. Предполагая, что атрибут a имеет V возможных значений, D'v означает, что значение атрибута a в D' является подмножеством выборки aVd, а D'k означает k-е в D '(k = 1,2, ..., | y |) Для выборки подмножества класса мы присваиваем вес wx каждой выборке. Определите следующие переменные[2]:

(1 ≤ k ≤ |y|)

(1 ≤ v ≤ V)

А формула получения информации расширяется до следующего вида:

Gain(D, a) = p \* Gain()

= p \* (Ent() - )

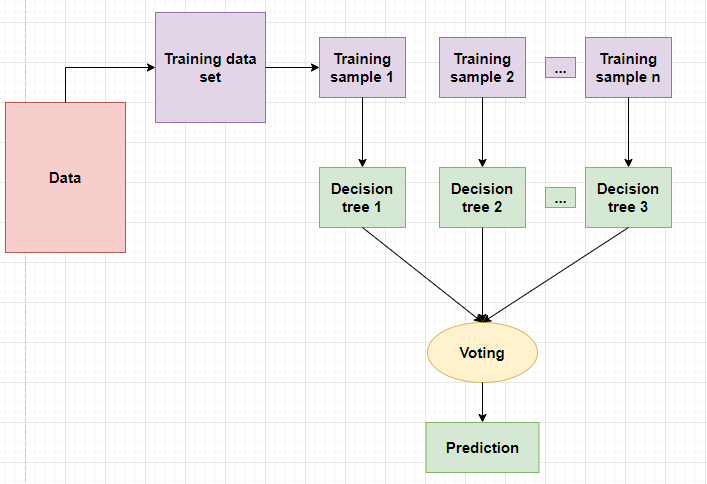
Среди них:

Ent() = -

* + 1. **Методы построения деревьев**

Существует несколько популярных методов построения деревьев:

* ID3: использует энтропийный критерий. Строит дерево до тех пор, пока в каждом листе не окажутся объекты одного класса, либо пока разбиение вершины дает уменьшение энтропийного критерия.
* C4.5: использует критерий Gain Ratio (нормированный энтропийный критерий). Критерий останова — ограничение на число объектов в листе. Обработка пропущенных значений осуществляется с помощью метода, который игнорирует объекты с пропущенными значениями при вычислении критерия ветвления, а затем переносит такие объекты в оба поддерева с определенными весами.
* CART: использует критерий Gini.
  1. **Алгоритм случайного леса**



*Рис.4 – Построения случайный лес на основе дерева решений.*

Брейманом был предложен метод, известный как случайные леса. Фактически случайный лес Бреймана - это ансамбль деревьев решений, каждое из которых строится на основе бутстреп выборки из исходной обучающей выборки (баггинг) , причем для расщепления вершин аналогично используется только доля случайно отбираемых признаков. Кроме того, строится полное дерево (без усечения). Классификация деревьев в ансамбле осуществляется большинством голосов [3].

Алгоритм индукции случайного леса может быть представлен в следующем виде[4]:

* Для i = 1, 2, . . . , B (здесь B — количество деревьев в ансамбле) выполнить
* Сформировать бутстреп выборку S размера l по исходной обучающей выборке D = {xi, yi}li=1
* По бутстреп выборке S индуцировать неусеченное дерево решений Ti с минимальным количеством наблюдений в терминальных вершинах равным nmin, рекурсивно следуя следующему подалгоритму:
* из исходного набора n признаков случайно выбрать p признаков
* из p признаков выбрать признак, который обеспечивает наилучшее расщепление
* расщепить выборку, соответствующую обрабатываемой вершине, на две подвыборки
* В результате выполнения шага 1 получаем ансамбль деревьев решений {Ti}Bi=1
* Для классификации, предсказание новых наблюдений осуществлять следующим образом:
* Пусть - класс, предсказанный деревом решений Ti, т.е. Ti(x) = ; тогда - класс, наиболее часто встречающийся в множестве {}Bi=1.

Одно из достоинств случайных лесов состоит в том, что для оценки вероятности ошибочной классификации нет необходимости использовать кросс–проверку или тестовую выборку. Оценка вероятности ошибочной классификации случайного леса осуществляется методом "Out–Of–Bag" (OOB) [5]. Известно, что каждая бутстреп выборка не содержит примерно 37 % наблюдений исходной обучающей выборки (поскольку выборка с возвращением, то некоторые наблюдения в нее не попадают, а некоторые попадают несколько раз). Классифицируем некоторый вектор x ∈ D. Для классификации используются только те деревья случайного леса, которые строились по бутстреп выборкам, не содержащим x, и как обычно используется метод голосования. Частота ошибочно классифицированных векторов обучающей выборки при таком способе классификации и представляет собой оценку вероятности ошибочной классификации случайного леса методом OOB. Практика применения оценки OOB показала, что в случае, если количество деревьев достаточно велико, эта оценка обладает высокой точностью. Если число деревьев мало, то оценка имеет положительное смещение [6].

### 1.3.1. Out–Of–Bag

В идеальном случае около 36,8 % всех обучающих данных составляют выборку OOB. Это можно показать следующим образом:

Если в наборе данных есть N строк. Тогда вероятность не выбрать строку при случайном выборке равна:

Вероятность не выбрать N строк при случайном выборке равна:

**1.3.2. Случайность случайного леса**

На случайность влияют два фактора:

* Случайность выборки - каждое дерево решений строится из случайного набора данных, извлеченных из исходных данных.
* Случайность атрибутов - при построении каждого дерева решений случайным образом выбирается определенное количество атрибутов, и наиболее подходящий атрибут выбирается в качестве узла разделения.

**1.3.3. Случайный лес в задачах регрессии и классификации**

Для задачи классификации ответ дается по большинству.

Для задачи регрессии результат дается как среднее значение деревьев решений.

**1.3.4. Преимущества алгоритма случайного леса**

* Он преодолевает проблему переоснащения путем усреднения или объединения результатов различных деревьев решений.
* Случайные леса хорошо работают с большим количеством элементов данных, чем одно дерево решений.
* Случайный лес имеет меньшую дисперсию, чем одно дерево решений.
* Алгоритмы Random Forest поддерживают хорошую точность даже при случае отсутствии части данных.

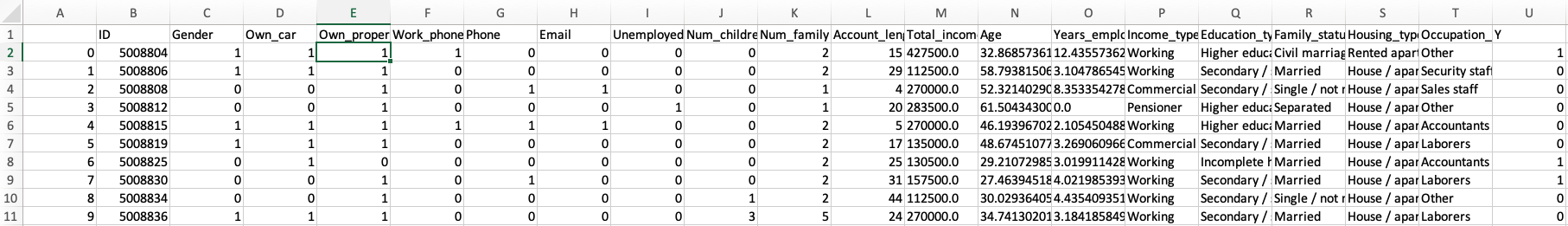
Недостатки алгоритма случайного леса:

* Сложность является основным недостатком алгоритмов случайного леса поэтому процесс прогнозирования с использованием случайных лесов очень трудоемкий по сравнению с другими алгоритмами.
* Построение случайных лесов намного сложнее и отнимает больше времени, чем деревья решений.
  1. **Данные для построения и тестирования деревьев решений для классификации добросовестных соискателей кредита**

Для обучения моделей машинного обучения не менее важной задачей является поиск анализирование данных для обучения и тестирования модели. Источник данных должен быть достаточным по количеству и иметь отношение к нашей задаче. На основе атрибутов этих данных мы можем построить и уточнить модель машинного обучения.

В данном работе, для применение алгоритма случайного леса с целью исключения недобросовестных соискателей в процессе одобрения кредита, использование источник данных из ‘*Simplifying decision trees (1987) J. R. Quinlan, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545, Technology Square, Cambridge, MA 02139, U.S.A.*’ [7]. Данные обнародуются на сайте:

<https://www.kaggle.com/datasets/samuelcortinhas/credit-card-classification-clean-data>



*Рис.5 - Данные в .csv файле.*

* + 1. **Структура данных**

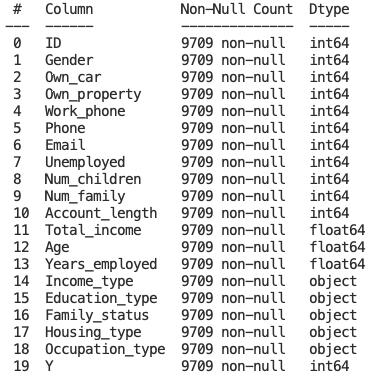
**Строки:**

* 1 - Имена атрибутов, и переменная ответа (Y).
* От 2 до конца – данные.

**Столбцы:**

* ID – Индекс клиента.
* Gender: Пол клиента (1 = мужской; 2 = женский).
* Own\_car: Есть ли у клиента автомобиль? (1 = Да, 0 = Нет).
* Own\_property: Есть ли у клиента собственность? (1 = Да, 0 = Нет).
* Work\_phone: Есть ли у клиента рабочий телефон? (1 = Да, 0 = Нет).
* Phone: Есть ли у клиента телефон? (1 = Да, 0 = Нет).
* Email: Есть ли у клиента адрес электронной почты? (1 = Да, 0 = Нет).
* Unemplpoyed: Клиент безработный? (1 = Да, 0 = Нет).
* Num\_children: Число детей.
* Num\_family: Количество членов семьи.
* Account\_length: Количество владения кредитной картой.
* Total\_income: Общий доход (китайские юани).
* Age: Лет.
* Years\_employed: Количество лет работы.
* Income\_type: Тип дохода.
* Education\_type: Тип образования.
* Family\_status: Брак статус.
* Housing\_type: Тип корпуса.
* Occupation\_type: Род занятий.
* Target: Цель (1 = высокий риск, 0 = низкий риск).

Данные состоят из 9710 строк.

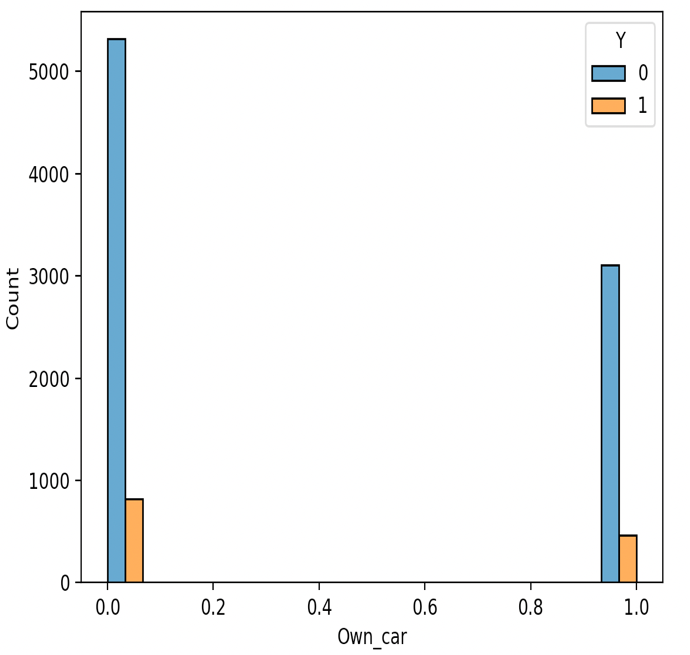
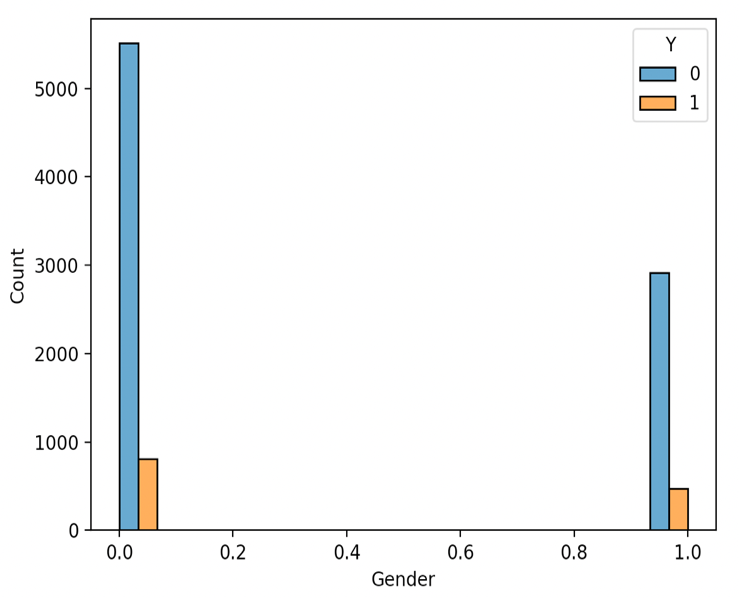


*Рис.6 - Информация о данных.*

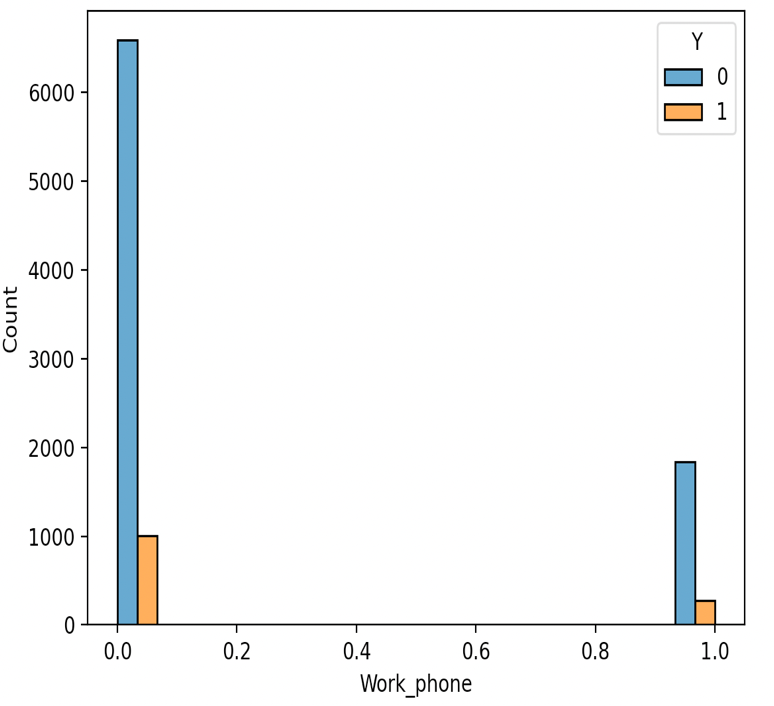
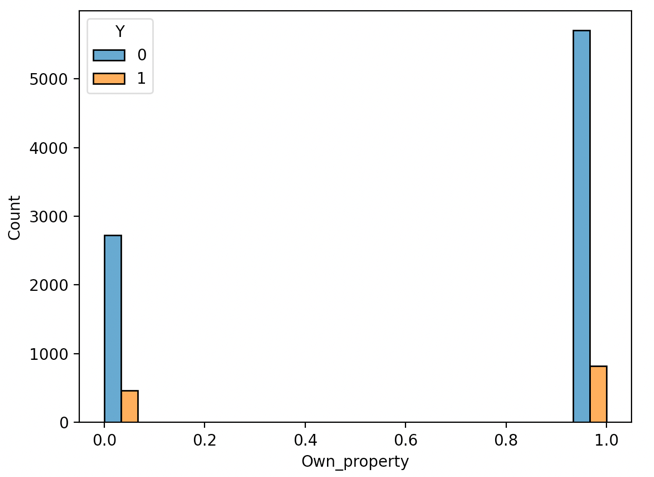
* + 1. **Анализ отношений между переменными и метками**

Машинное обучение тесно связано с данными. Данные сильно влияют на процесс обучения модели, включая размер данных, количество атрибутов, распределение значений в одном атрибуте и т. д. Однако машинное обучение - это не большие данные, и наоборот. Так что не анализировать все аспекты данных, а только анализировать факторы влияющие на дисбаланс распределение данных.

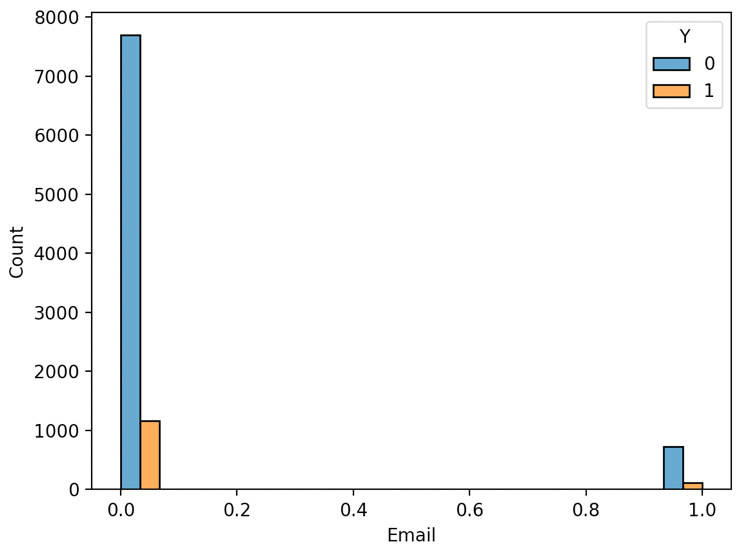
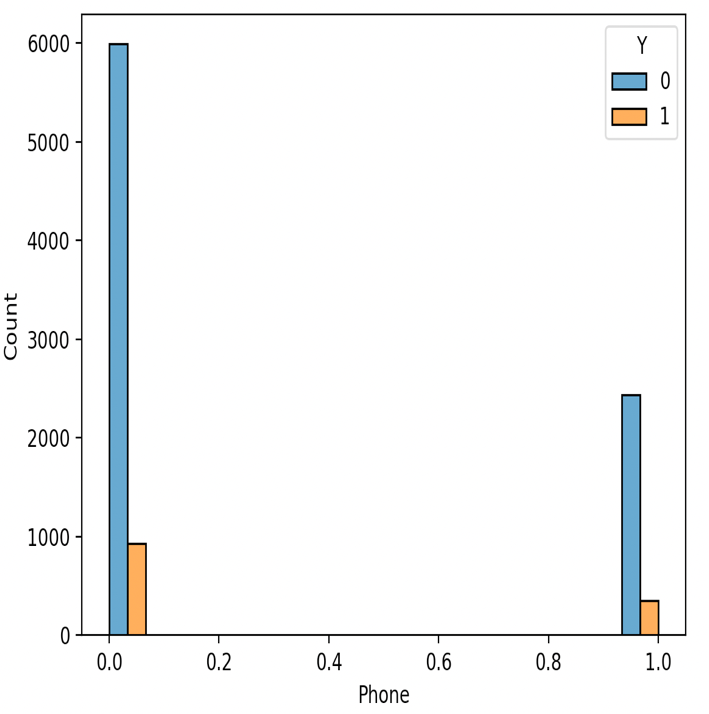
* + - 1. **Числовые переменные**

****

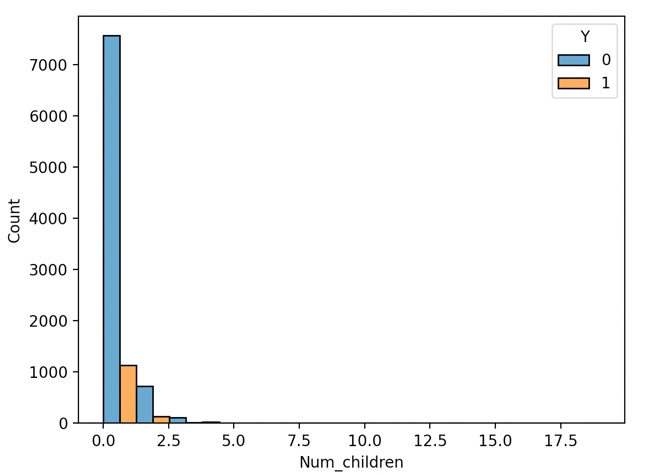
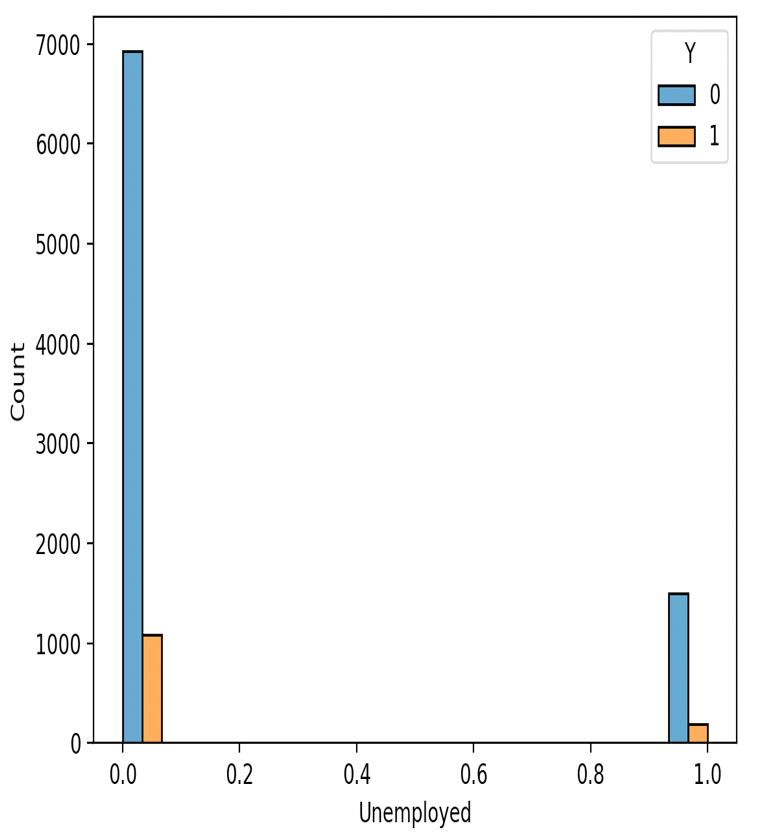
*Рис.7 - Количество правильных и неправильных ответов (меток) по значениям атрибута Gender и Own\_car.*

****

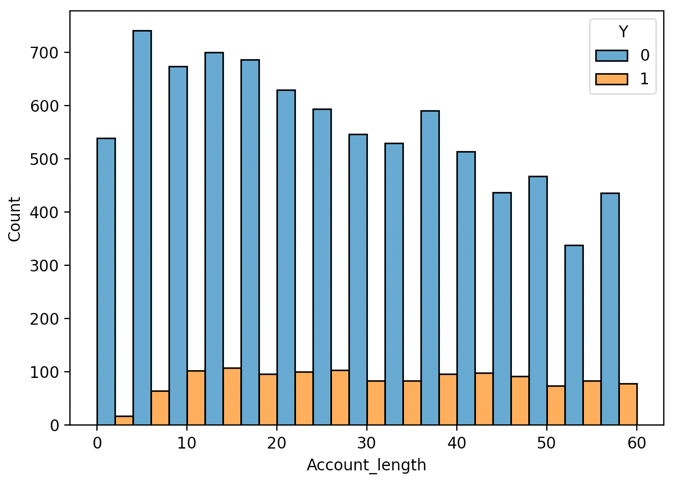
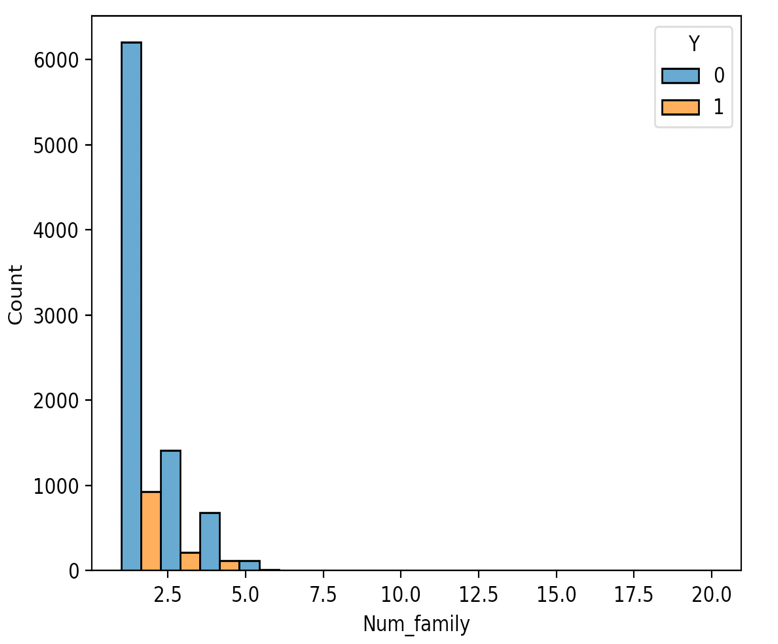
*Рис.8 - Количество правильных и неправильных ответов (меток) по значениям атрибута Own\_property и Work\_phone.*

****

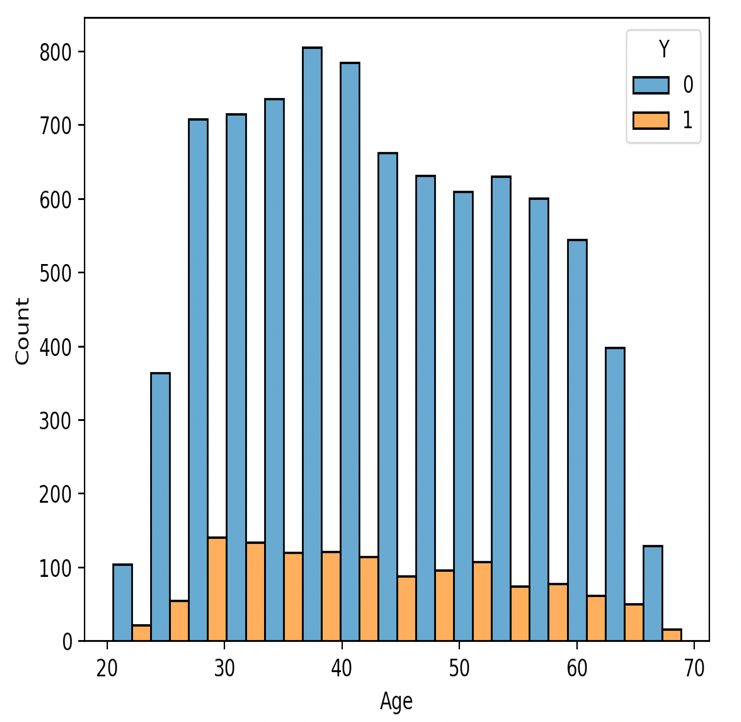
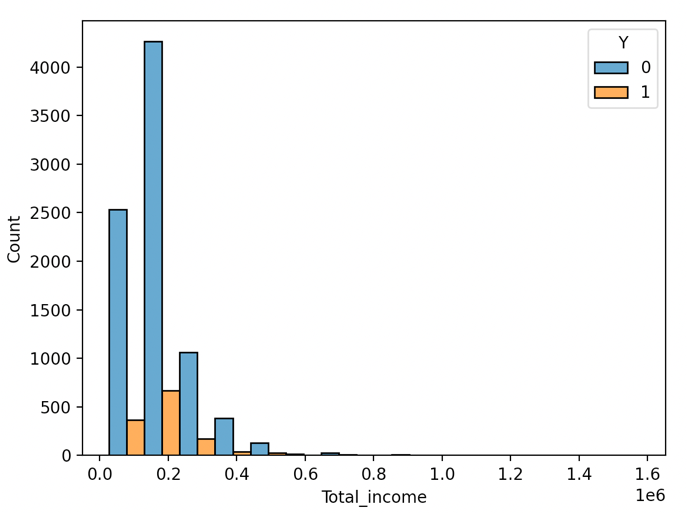
*Рис.9 -* *Количество правильных и неправильных ответов (меток) по значениям атрибута Phone и Email.*

****

*Рис.10 - Количество правильных и неправильных ответов (меток) по значениям атрибута Phone и Email.*

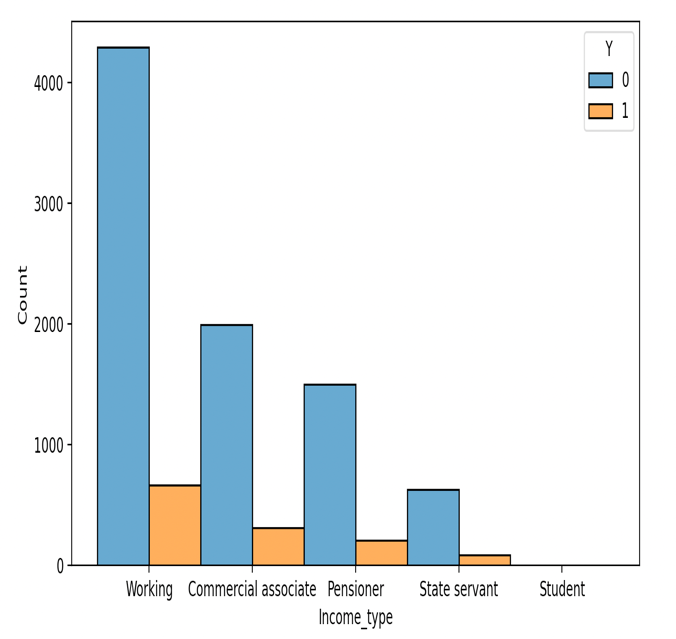
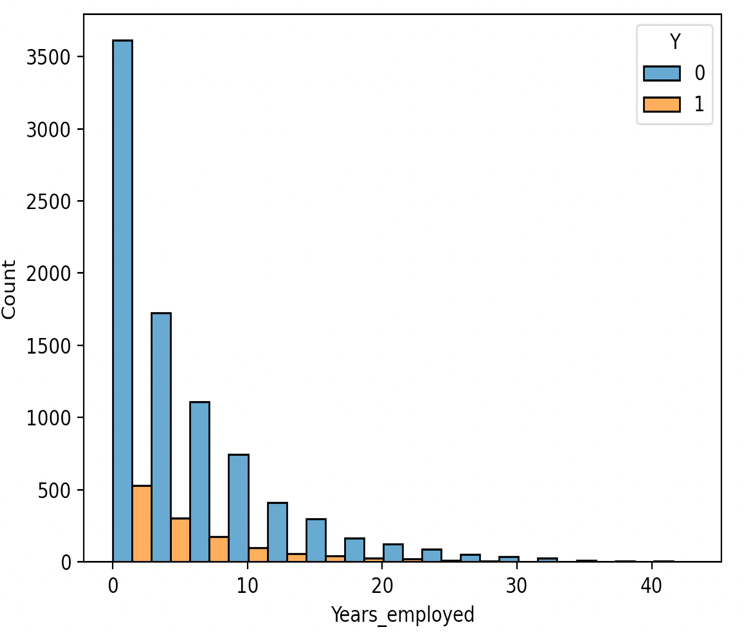
****

*Рис.11 - Количество правильных и неправильных ответов (меток) по значениям атрибута Num\_family и Account\_length.*

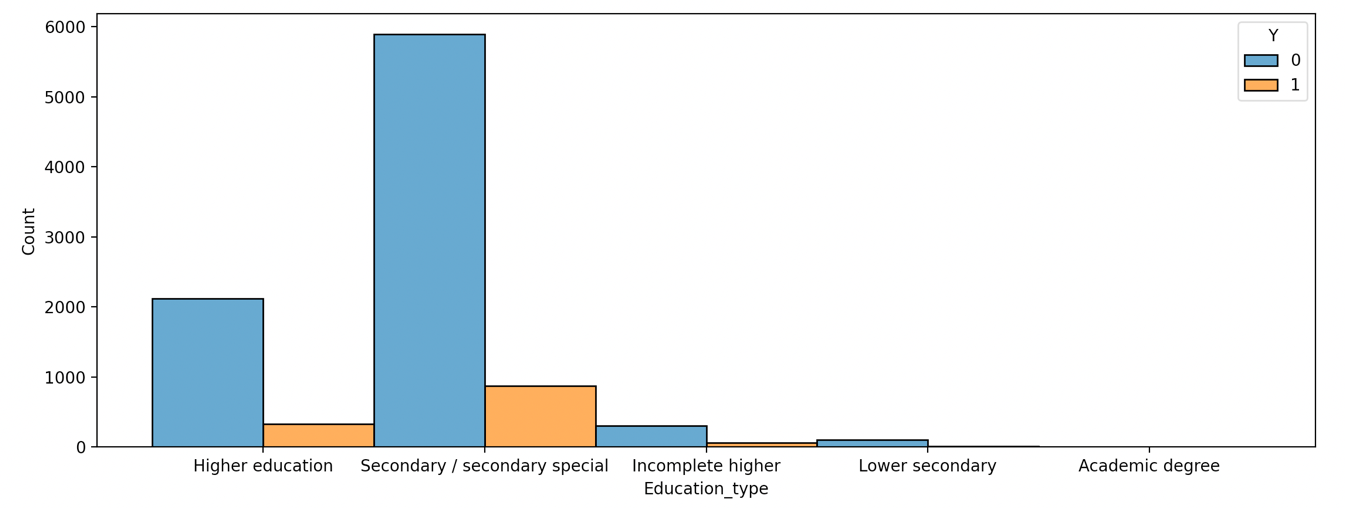
****

*Рис12 - Количество правильных и неправильных ответов (меток) по значениям атрибута Total\_income и Age.*

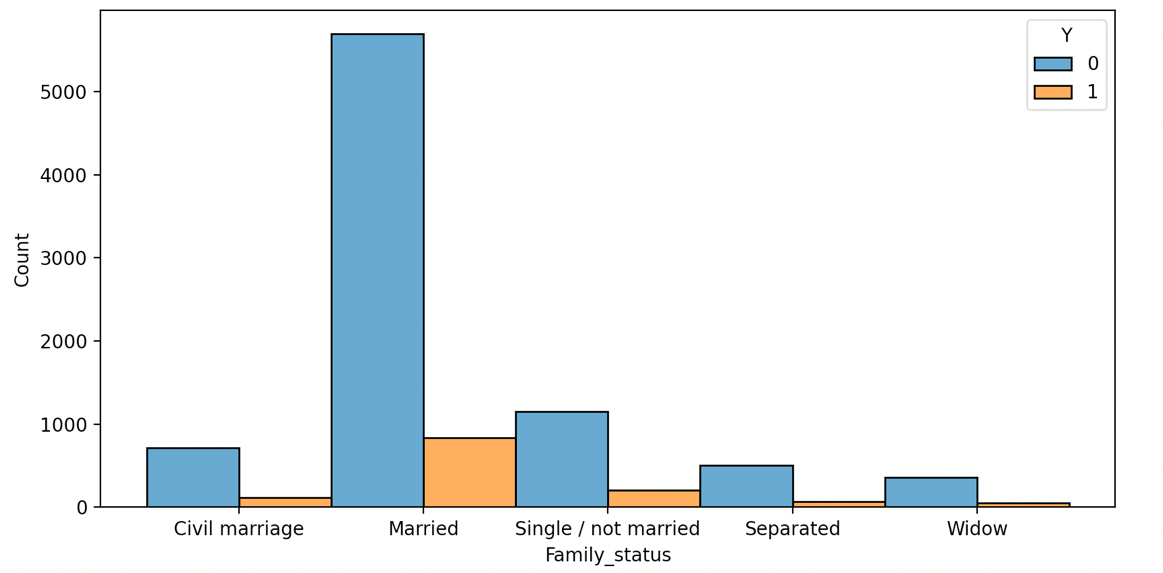
* + - 1. **Категориальные переменные**

****

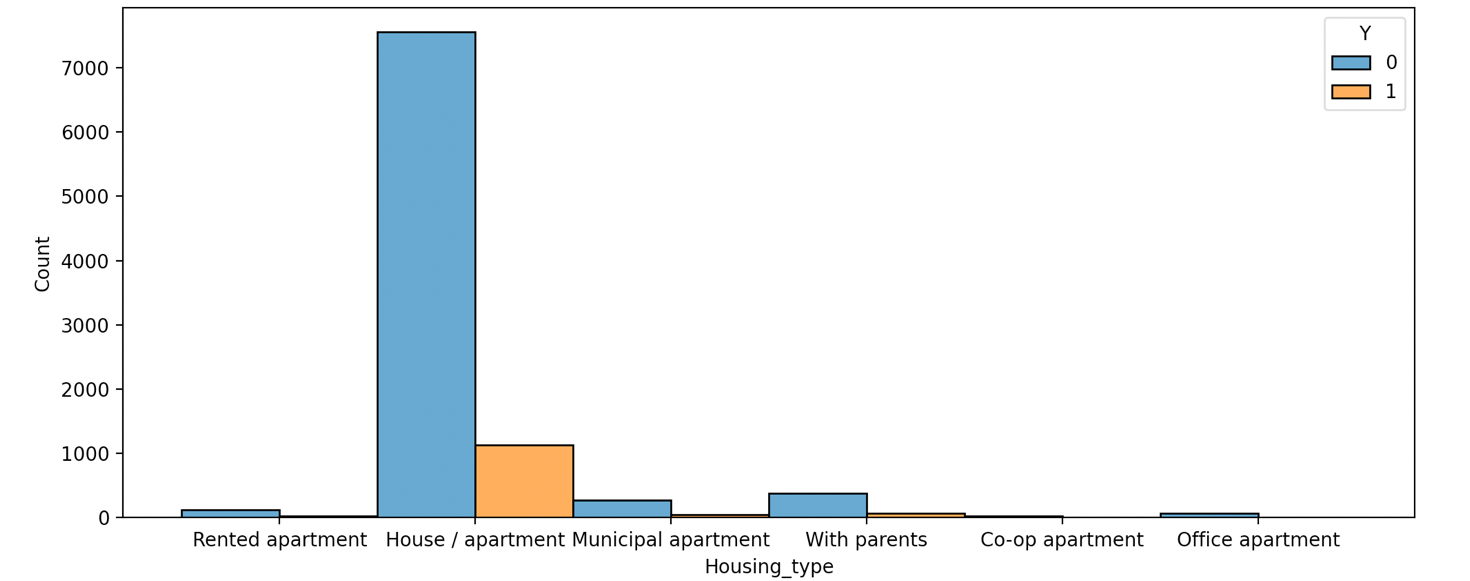
*Рис.13 - Количество правильных и неправильных ответов (меток) по значениям атрибута Years\_employed и Income\_type.*

****

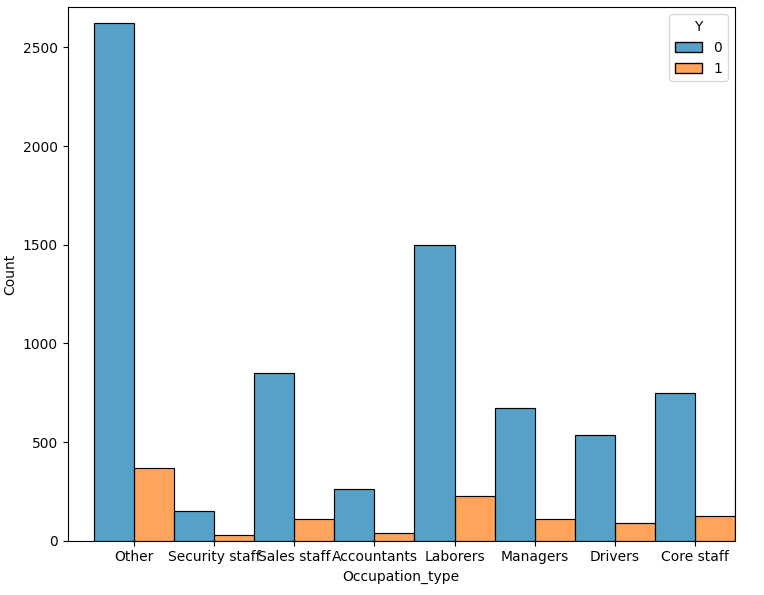
*Рис.14 - Количество правильных и неправильных ответов (меток) по значениям атрибута Education\_type*.

****

*Рис.15 - Количество правильных и неправильных ответов (меток) по значениям атрибута Family\_status.*

****

*Рис.16 - Количество правильных и неправильных ответов (меток) по значениям атрибута Housing\_type.*

****

*Рис.17 - Количество правильных и неправильных ответов (меток) по значениям атрибута Occupation\_type.*

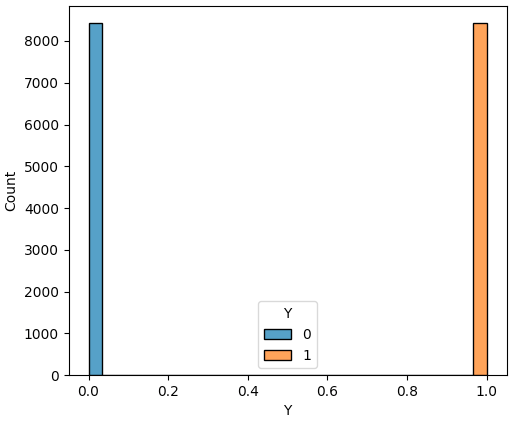
На рисунках с 7 по 17 значение метки Y (ответ), количество ответов равно 0, всегда составляет большинство всех переменных, около 87%. Это предполагает, что модель может быть ответить, на всех испытание 0, и, очевидно, она получит точность 87%. Чтобы решить этот проблем, есть методы: избыточная выборка, недостаточная выборка, …

* + 1. **Дисбаланс распределения категорий выборки**

Подготовка данных — невероятно важный шаг в процессе машинного обучения.

### 1.4.3.1. Избыточная выборка

Копирование данных с небольшим размером выборки и добавьте их в выборку, чтобы добиться максимально возможного баланса между положительными и отрицательными выборками. После применения этого метода был достигнут баланс в данных (рис. 18).



*Рис.18 - Количество правильных и неправильных ответов (меток).*

### 1.4.3.2. Недостаточная выборка

Недостаточная выборка - удаление данных из данных, метки которых преобладают над остальными метками. После применения этого метода был достигнут баланс в данных (рис. 19).



*Рис.19 - Количество правильных и неправильных ответов (меток).*

* 1. **Оценить эффективность модели машинного обучения**

Для определения эффективности моделей машинного обучения необходимо показать следующие: матрица путаницы (confusion matrix), точность(accuracy), …

* + 1. **Матрица путаницы – confusion matrix**

|  |  |  |
| --- | --- | --- |
| Настоящая категория | Категория прогноза | |
| Положительный пример | Отрицательный пример |
| Положительный пример | TP | FN |
| Отрицательный пример | FP | TN |

*Таб.2 - Матрица путаницы.*

Точность (Precision):

Precision в идеале должна быть 1 (высокая) для хорошего классификатора. Точность становится равной 1 только тогда, когда числитель и знаменатель равны, т. е. TP = TP + FP, это также означает, что FP равен нулю. Когда увеличения FP значение знаменателя становится больше числителя, а значение точности уменьшается.

Скорость отзыва (Recall):

В идеале Recall должна быть 1 (высокая) для хорошего классификатора. Recall становится равным 1 только тогда, когда числитель и знаменатель равны, т. е. TP = TP + FN, это также означает, что FN равен нулю. Когда увеличения FN значение знаменателя становится больше числителя, а значение отзыва уменьшается.

Значение F1:

Показатель F1 становится равным 1 только тогда, когда и precision, и recall равны 1. Показатель F1 становится высоким только тогда, когда и precision, и recall высоки. Показатель F1 является гармоническим средним значением precision и recall и является лучшим показателем, чем accuracy.

* + 1. **Точность - accuracy**

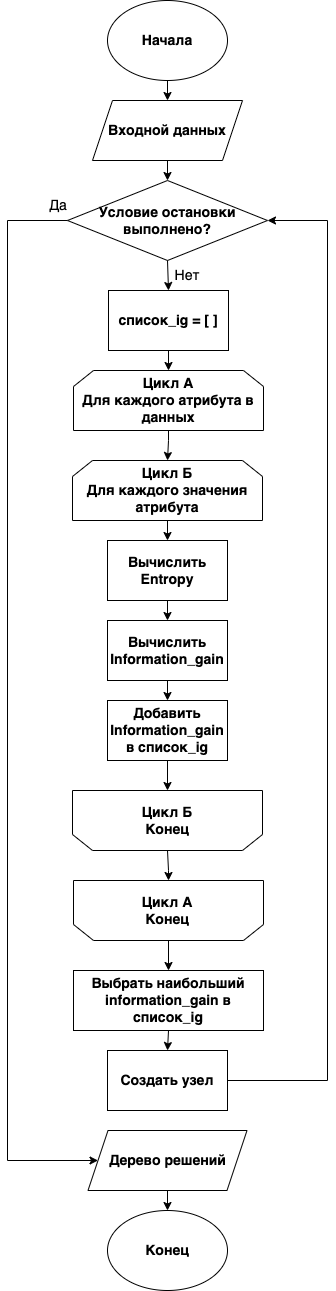
Точность – accuracy для быстро оценить общую точность алгоритма. Однако есть много недостатков, особенно в случае несбалансированных данных.

# Вывод

Проанализировали понятие о деревьях решений, тем самым поняв, как построить алгоритм дерева решений, а также его преимущества и недостатки. Как основа для построения алгоритма случайного леса, его преимущества и недостатки. Кроме того, также нашли правильные данные для Применение алгоритма случайного леса для машинного обучения при классификации добросовестных соискателей кредита.

1. **Конструкторский раздел**
   1. **Описание алгоритма дерева решений**
      1. **Схема алгоритма**

Схема алгоритма дерева решений показана на рисунке 20.

****

*Рис.20 - Схема алгоритма* *дерева решений.*

* + 1. **Описание алгоритма**

В данном работе, алгоритм реализован в рекурсивной форме, подробно описанной следующим образом:

Функция **построение**(*данные, имя метки столбца, примесная функция, максимальная глубина, минимальное разделение выборки, переменную счетчика глубины*):

1. Проверить условие остановки, если да, перейти к шагу i, наоборот:
   1. Выбор атрибуты и значения для разделения.
   2. Разделить данные на 2 части.
   3. Если деление удалось, перейти к шагу d, наоборот:
      1. Возвращение наиболее распространенное значение в метке в качестве ответа.
   4. Вызывать функция **построение**(*Первая часть данных,…*)
   5. Вызывать функция **построение**(*Вторая часть данных,…*)
   6. Сохранить свойства узла.
2. Возвращение наиболее распространенное значение в метке в качестве ответа.

Каждый узел дерева решений описывается следующим:

Листинг 1. Узел дерева решений.

class Node:

def \_\_init\_\_(self, feature=None, threshold=None, data\_left=None, data\_right=None, gain=None, value=None):

self.feature = feature

self.threshold = threshold

self.data\_left = data\_left

self.data\_right = data\_right

self.gain = gain

self.value = value

В котором:

Feature - Имя атрибута, выбранного для разделения.

Threshold - Значение в точке деления.

Data\_left - Левая часть родительских данных.

Data\_right - Правая часть родительских данных.

Gain - Прирост информации.

Value – Ответ, из наиболее значение в метке в качестве ответа.

* 1. **Описание алгоритма случайного леса**
     1. **Схема алгоритма**

На основе построенного дерева решений оставшаяся работа по построению случайного леса состоит в том, чтобы выбрать выборки данных, а затем построить дерево решений на этих выборках. Подробно показано на рисунке 21.

****

*Рис.21 - Схема алгоритма* *случайного леса.*

* + 1. **Описание алгоритма**

1. Выбрать M случайные наборы данных из обучающих данных.

2. Для каждого x в M:

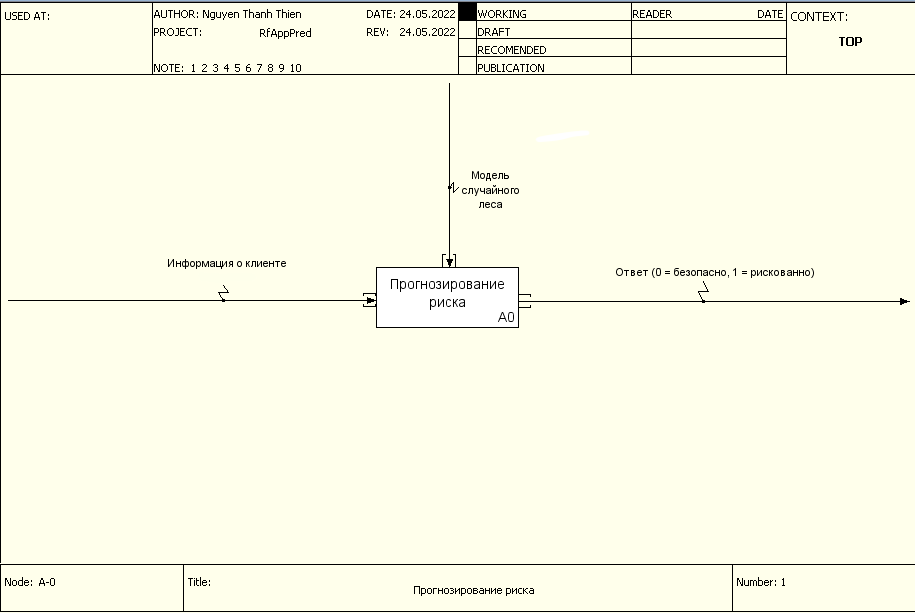
a. Построить дерево решений из х.

b. Добавить дерево решений в список деревьев решений случайного леса.

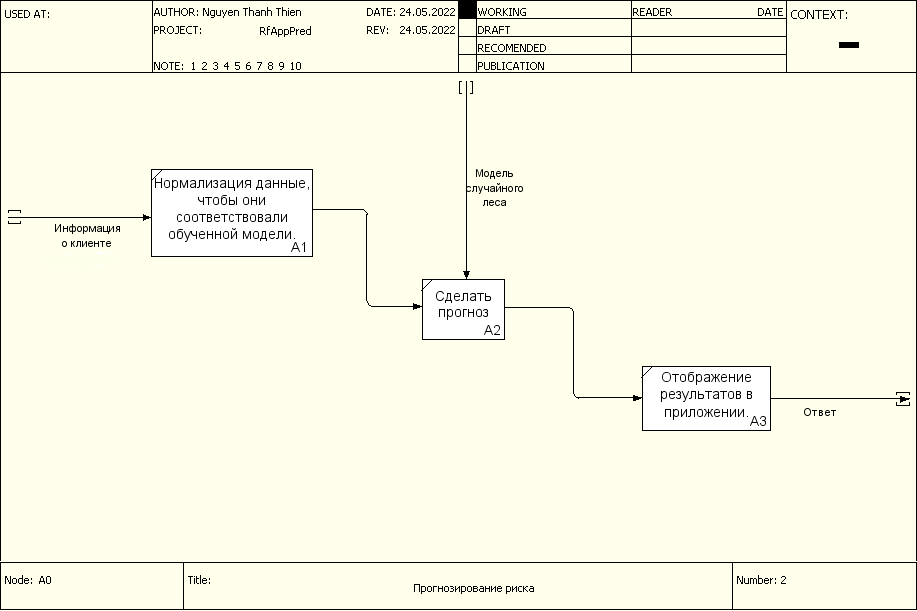
3. Сохранить модель случайного леса.

* 1. **Приложение для прогнозирования рисков**

Диаграмма приложения IDEF0 изображена на рис. 22 и 23.

**

*Рис.22 – IDEF0 приложения для прогнозирования рисков клиента, 1-й уровень.*

**

*Рис.23 – IDEF0 приложения для прогнозирования рисков клиента, 2-й уровень.*

# Вывод

В данном раздели, выполнил описание конструкции алгоритма случайного леса, основанного на алгоритме дерева решений. Выполнил описание конструкции приложение для прогнозирования рисков клиента. Показано с помощью логических диаграмм и диаграмм IDEF0.

1. **Технологический раздел**
   1. **Язык программирования**

Python 3.9 был выбран для разработки проекта.

Python — это язык программирования высокого уровня, который легко изучить и понять. Также есть множество библиотек, помогающих с обработкой данных (pandas, numpy, …). Несмотря на недостаток медленной скорости выполнения, в настоящее время Python является широко используемым языком в таких областях, как наука о данных и искусственный интеллект, …

* 1. **Редактор кода**

Visual studio code - позиционируется как «лёгкий» редактор кода. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом.

* 1. **Подготовка набора данных**
     1. **Загрузить данные в программу**

Данные сохраняются в виде файла .csv, загружено с поддержкой библиотеки pandas.

Листинг 2. Загрузить данные в программу.

import pandas as pd

df = pd.read\_table("./data/clean\_data.csv", delimiter=',')

* + 1. **Обработка дисбаланса данных**

В данном работе, используя метод избыточного выборки для отработку дисбаланса данных.

Листинг 3. Функция обработчика дисбаланса данных.

def oversample(df):

classes = df.Y.value\_counts().to\_dict()

most\_value = max(classes.values())

classes\_list = []

for cl in classes:

classes\_list.append(df[df['Y'] == cl])

classes\_sample = []

for i in range(1,len(classes\_list)):

classes\_sample.append(classes\_list[i].sample(most\_value, replace = True))

df\_maybe = pd.concat(classes\_sample)

result = pd.concat([df\_maybe, classes\_list[0]], axis=0)

result = result.reset\_index(drop = True)

return result

* + 1. **Смешивание и разделение данных**

После обработки данных о дисбалансе, данные содержат в общей 16852 строки. Использовать библиотеку sklearn для смешивать данные, затем использовать 80% данных для обучения модели и 20% для проверки результатов.

Листинг 4. Смешивание и разделение данных.

from sklearn.utils import shuffle

df = shuffle(df)

df\_train = df[:13481]

df\_test = df[13481:]

* 1. **Вычислить энтропии и прирост информации**

Для построить дерево методом ID3, энтропии и прирост информации используются вместо индекса Джини.

Листинг 5. Вычислить энтропии.

def entropy(var):

try:

a = var.value\_counts() / var.shape[0]

entropy = np.sum(-a \* np.log2(a))

return(entropy)

except Exception as e:

print(e)

Листинг 6. Вычислить прирост информации.

def information\_gain(target\_var, mask, func):

a = sum(mask)

b = mask.shape[0] - a

if (a == 0 or b ==0):

ig = 0

else:

#for classification

ig = func(target\_var) - a / (a + b) \* func(target\_var[mask]) - b / (a + b) \* func(target\_var[-mask])

return ig

* 1. **Построение дерева решений**

Дерево решений строится рекурсивно для каждого узла дерева.

Листинг 7. Построение дерева решений.

def train\_tree(data, y, impurity\_func = entropy, max\_depth = 5,min\_samples\_split = 10, counter=0):

if len(data) >= min\_samples\_split and counter <= max\_depth:

variable, max\_best\_ig, split\_value = get\_best\_option(data, y, func = impurity\_func)

if max\_best\_ig is not None:

left, right = make\_split(variable, split\_value, data)

if left.empty or right.empty:

return Node(value=Counter(data[y]).most\_common(1)[0][0])

yes\_answer = train\_tree(left, y, impurity\_func, max\_depth, min\_samples\_split, counter + 1)

no\_answer = train\_tree(right, y, impurity\_func, max\_depth, min\_samples\_split, counter + 1)

return Node(

feature = variable,

threshold = split\_value,

data\_left = yes\_answer,

data\_right = no\_answer,

gain = max\_best\_ig)

return Node(value=Counter(data[y]).most\_common(1)[0][0])

* 1. **Случайный лес класса**

Класс случайного леса включает в себя: построение случайного леса, тестирование производительности модели, создание прогнозов.

Листинг 8. Случайный лес класса.

class RandomForest:

def \_\_init\_\_(self, data, impurity\_func = entropy, max\_depth = 20,min\_samples\_split = 2, trees = 25):

self.data = data

self.y = str(data.columns[-1])

self.impurity\_func = impurity\_func

self.max\_depth = max\_depth

self.min\_samples\_split = min\_samples\_split

self.trees = trees

self.list\_tree = []

def create\_fores(self):

#Split data to any samples

data = self.data.copy()

list\_samples = []

size = (len(data) \* 80) // 100

for i in range(0, self.trees):

list\_samples.append(data.sample(size))

#create trees

t = 0

for df in list\_samples:

tree = train\_tree(df, self.y, self.impurity\_func, self.max\_depth, self.min\_samples\_split, 0)

t += 1

print('Tree {} was created'.format(t))

self.list\_tree.append(tree)

def test\_forest(self, df\_test):

list\_pred = []

total\_0 = (df\_test[str(df\_test.columns[-1])] == 0).sum()

total\_1 = (df\_test[str(df\_test.columns[-1])] == 1).sum()

print(total\_0, total\_1)

for i in range(len(df\_test)):

list\_result = []

for tree in self.list\_tree:

list\_result.append(int(prediction(tree, df\_test.iloc[i])))

counter = Counter(list\_result)

list\_pred.append(counter.most\_common(1)[0][0])

return list\_pred

def take\_prediction(self, df\_row):

list\_result = []

for tree in self.list\_tree:

list\_result.append(int(prediction(tree, df\_row)))

counter = Counter(list\_result)

return counter.most\_common(1)[0][0]

* 1. **Сохранение и загрузка случайной леса модели**

Обучение модели машинного обучения обычно занимает много времени. Поэтому необходимо сохранить обученную модель для использования в прогнозировании. Для этого, используется библиотека pickle.

Листинг 9. Сохранение и загрузка случайной леса модели.

import pickle

#save model

with open('./model/forest.model1', 'wb') as randomforestmodel:

pickle.dump(randomforest, randomforestmodel)

#load model

with open('./model/forest.model1', 'rb') as randomforestmodel:

my\_forest = pickle.load(randomforestmodel)

* 1. **Оценка эффективности обученной модели**

Для оценки модели вычисляется точности с помощью библиотеки sklearn. Матрица путаницы, нарисованная с помощью библиотеки seaborn.

Листинг 10. Оценка эффективности обученной модели.

with open('./model/forest.model1', 'rb') as randomforestmodel:

my\_forest = pickle.load(randomforestmodel)

t2 = time.time()

y\_pred = my\_forest.test\_forest(df\_test)

print("The testing was done in {} seconds.".format(time.time() - t2))

y\_test = df\_test['Y'].to\_numpy()

y\_pred = np.array(y\_pred).astype(int)

print("Accuracy: {}%".format(accuracy\_score(y\_test, y\_pred) \* 100))

confusion\_matrix = pd.crosstab(y\_test, y\_pred, rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion\_matrix, annot=True)

plt.show()

* 1. **Пользовательский интерфейс**

Пользовательский интерфейс построен с поддержкой библиотеки pyqt5, после разработки пользовательского интерфейса, pyqt5 автоматически сгенерирует исходный код в соответствии с этим дизайном.

Однако для обработки взаимодействия между пользователем и интерфейсом написан следующий код.

Листинг 11.1 Пользовательский интерфейс.

class mywindow(QMainWindow):

def \_\_init\_\_(self):

super(mywindow, self).\_\_init\_\_()

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.ui.btn\_check.clicked.connect(self.onCheckBtnClick)

with open('./model/forest.model1', 'rb') as randomforestmodel:

self.my\_forest = pickle.load(randomforestmodel)

def mapping(self, var, value):

…

def onCheckBtnClick(self):

try:

Gender = int(self.ui.gender.currentText())

Own\_car = int(self.ui.car.currentText())

Own\_property = int(self.ui.property.currentText())

Work\_phone = int(self.ui.workphone.currentText())

Phone = int(self.ui.phone.currentText())

Email = int(self.ui.work\_phone.currentText())

Years\_employed = float(self.ui.year\_work.toPlainText())

Num\_children = int(self.ui.number\_childrens.toPlainText())

Num\_family = int(self.ui.number\_family.toPlainText())

Account\_length = int(self.ui.number\_card.toPlainText())

Total\_income = float(self.ui.total\_income.toPlainText())

Age = float(self.ui.yearsold.toPlainText())

Unemployed = int(self.ui.unemployee.currentText())

Income\_type = str(self.ui.type\_income.currentText())

Education\_type = str(self.ui.type\_education.currentText())

Family\_status = str(self.ui.famil\_status.currentText())

Housing\_type = str(self.ui.house\_type.currentText())

Occupation\_type = str(self.ui.work\_type.currentText())

except Exception as e:

msgBox = QMessageBox()

msgBox.setText(str(e))

Листинг 11.2 Пользовательский интерфейс.

msgBox.exec()

return

row = {'Gender' :Gender, 'Own\_car' :Own\_car, 'Own\_property' :Own\_property, 'Work\_phone' :Work\_phone, 'Phone' :Phone, 'Email' :Email, 'Unemployed' :Unemployed,

'Num\_children' :Num\_children, 'Num\_family' :Num\_family, 'Account\_length' :Account\_length, 'Total\_income' :Total\_income, 'Age' :Age,

'Years\_employed' :Years\_employed, 'Income\_type' :self.mapping('Income\_type', Income\_type),

'Education\_type' :self.mapping('Education\_type', Education\_type), 'Family\_status' :self.mapping('Family\_status', Family\_status),

'Housing\_type' :self.mapping('Housing\_type', Housing\_type), 'Occupation\_type' :self.mapping('Occupation\_type', Occupation\_type)}

try:

df = pd.DataFrame(row, index=[0])

result = self.my\_forest.take\_prediction(df.iloc[0])

if result == 0:

self.ui.lineEdit\_result.setStyleSheet("background-color: rgb(155, 242, 145);")

self.ui.lineEdit\_result.setText('Низкий риск :)')

else:

self.ui.lineEdit\_result.setStyleSheet("background-color: rgb(232, 95, 104);")

self.ui.lineEdit\_result.setText('Высокий риск :(')

except Exception as e:

msgBox = QMessageBox()

msgBox.setText(str(e))

msgBox.show()

msgBox.exec()

return

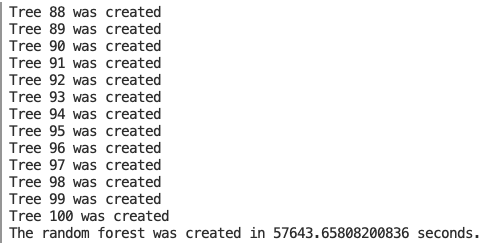
1. **Исследовательский раздел**
   1. **Описание экспериментов**

Обучение и тестирование случайного леса модели производилось на персональном ноутбуке со следующей конфигурацией:

* Операционная система: MacOS Big Sur.
* Процессор 1,4 GHz Quad-Core Intel Core i5.
* Оперативная память: 8GB 2133 MHz LPDDR3.

Использовать 80% данных (13481 строка) для обучения модели и 20% (3371 строка) чтобы оценить производительность модели. Через 57643 секунды (16 часов) была построена модель случайного леса, состоящая из 100 деревьев решений. Где каждое дерево решений построено на случайном наборе данных из 80% исходных обучающих данных (10784 строки). Параметры каждого дерева решений задаются следующим образом:

* Максимальная глубина: 20.
* Минимальное разделение выборки: 2 строки.

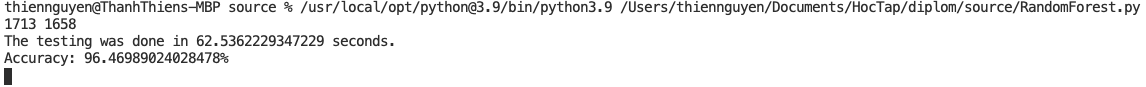


*Рис.24 – Лог-текст построения случайного леса.*

* 1. **Оценка эффективности модели**
     1. **Точность**

Точность модели: 96.47% из 3371 испытаний, включает 1713 положительных случая и 1658 отрицательных случаев.

Время выполнения экспериментов: 61 секунда.

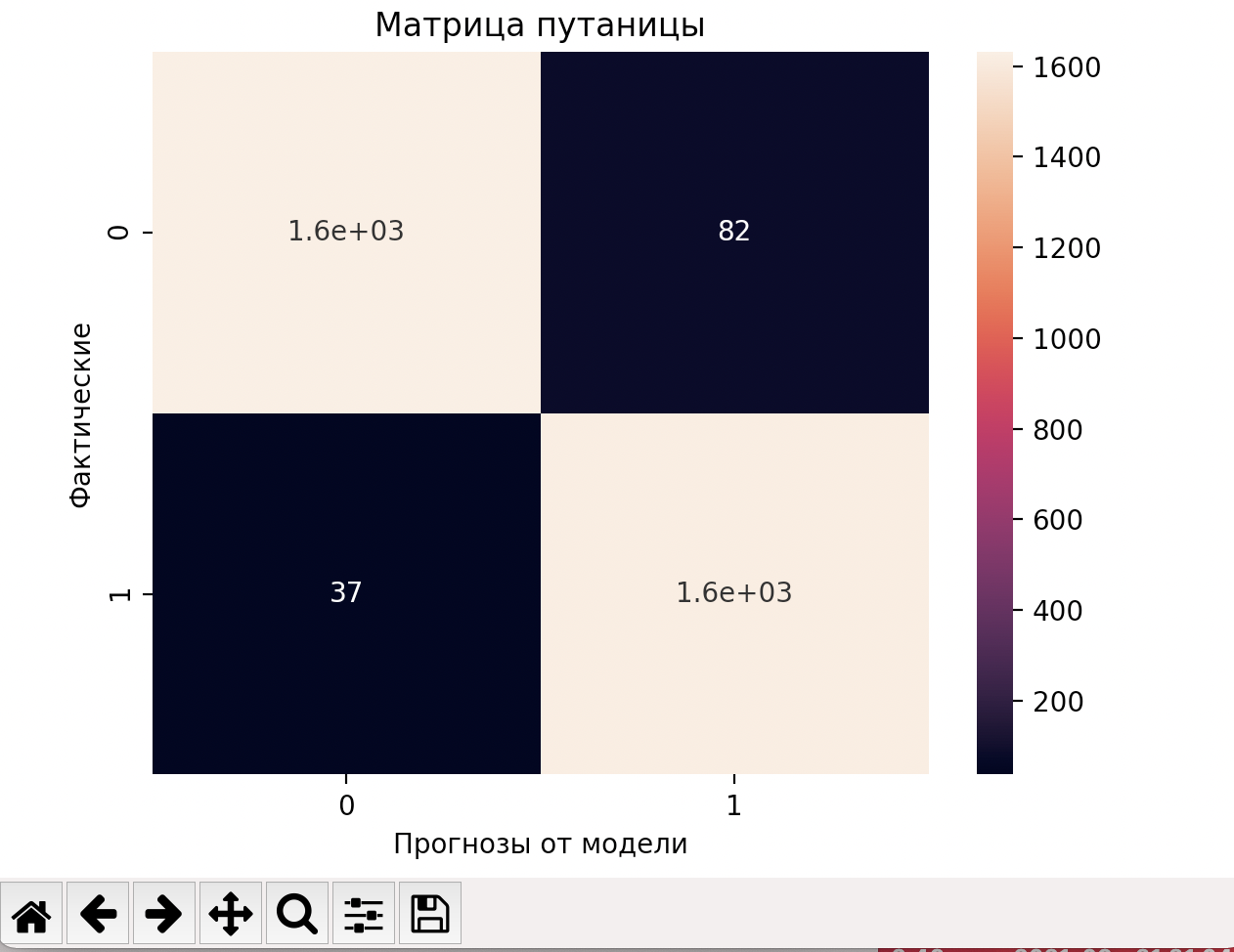


*Рис.25 – Лог-текст расчета точности модели.*

* + 1. **Матрица путаницы**

Всего проведено 3371 испытания, включает 1713 положительных случая и 1658 отрицательных случаев.

* Количество правильных ответов в случае положительного: 1626.
* Количество неправильных ответов в случае положительного: 82.
  + Точность в случае положительного: 95.2%
* Количество правильных ответов в случае отрицательного: 1626.
* Количество неправильных ответов в случае отрицательного: 37.
  + Точность в случае отрицательного: 97.8%.



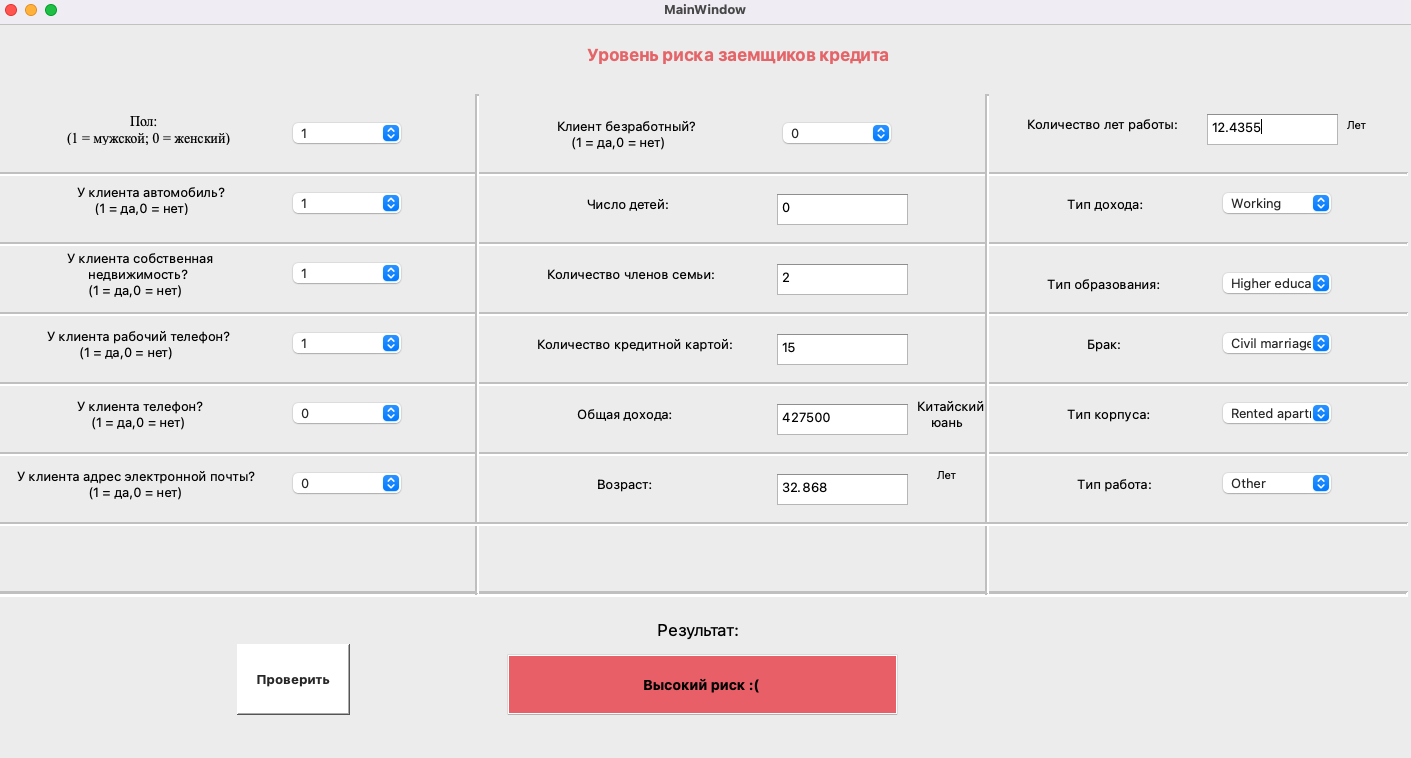
*Рис.26 – Матрица путаницы.*

* 1. **Приложение для прогнозирования риска соискателей в процессе одобрения кредита**
     1. **Прогнозирования соискатели с высоким риском**

Ввод информации о клиенте с высоким риском, извлеченной из тестовых данных:

* Пол клиента: 1.
* У клиента автомобиль: 1.
* У клиента собственность: 1.
* У клиента рабочий телефон: 1.
* У клиента телефон: 0.
* У клиента адрес электронной почты: 0.
* Клиент безработный: 0.
* Число детей: 0.
* Количество членов семьи: 2.
* Количество владения кредитной картой: 15.
* Общий доход (китайские юани): 427500.
* Лет: 32.9.
* Количество лет работы: 12.4.
* Тип дохода: Working.
* Тип образования: Higher education.
* Брак статус: Civil marriage.
* Тип корпуса: Rented apartment.
* Род занятий: Other.

Приложение делает прогноз, что это клиент с высоким риском, совпадая с результатами в исходных данных. Поле отображения результатов также становится красным.



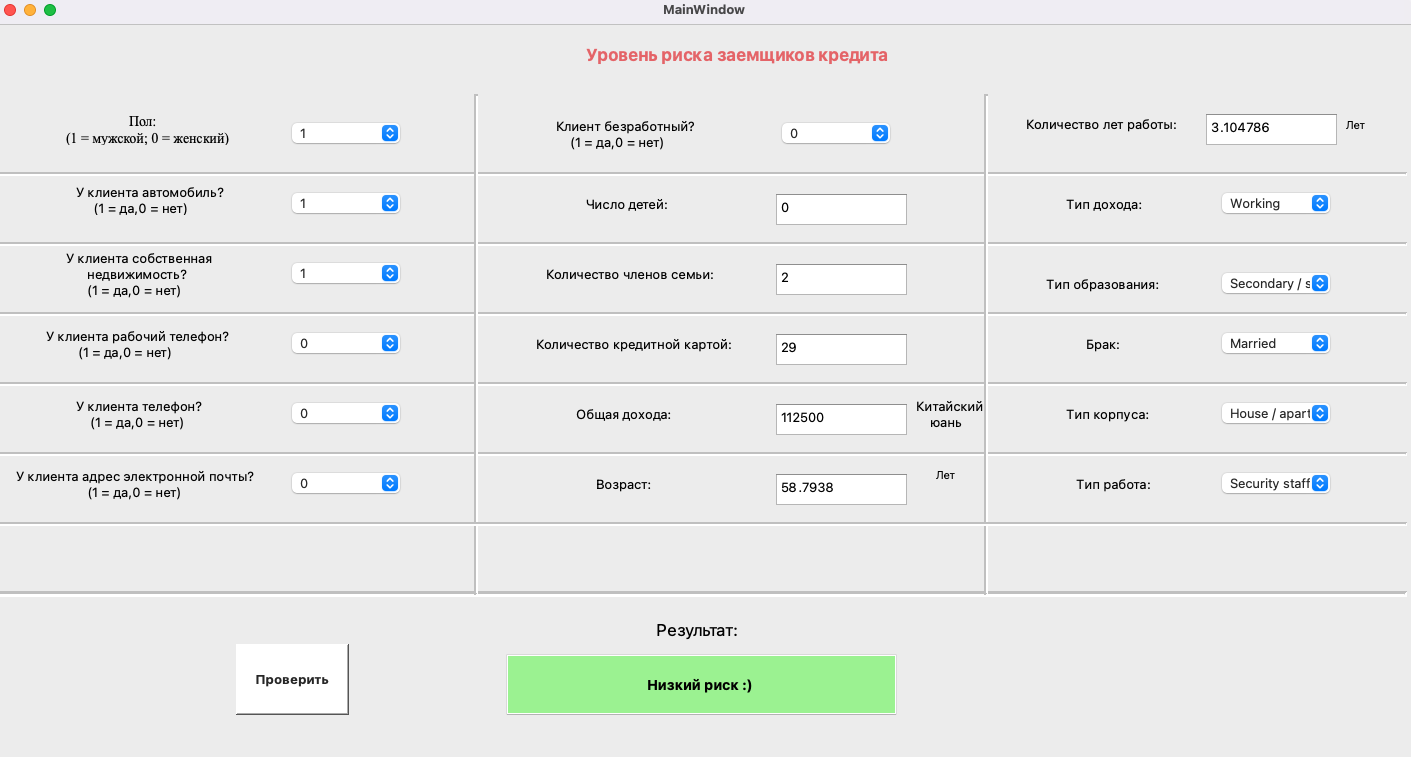
*Рис.27 – Прогнозирования соискатели с высоким риском.*

* + 1. **Прогнозирования соискатели с низким риском**

Ввод информации о клиенте с низким риском, извлеченной из тестовых данных:

* Пол клиента: 1.
* У клиента автомобиль: 1.
* У клиента собственность: 1.
* У клиента рабочий телефон: 0.
* У клиента телефон: 0.
* У клиента адрес электронной почты: 0.
* Клиент безработный: 0.
* Число детей: 0.
* Количество членов семьи: 2.
* Количество владения кредитной картой: 29.
* Общий доход (китайские юани): 112500.
* Лет: 58.8.
* Количество лет работы: 3.1.
* Тип дохода: Working.
* Тип образования: Secondary / secondary special.
* Брак статус: Married.
* Тип корпуса: House / apartment.
* Род занятий: Security staff.

Приложение делает прогноз, что это клиент с низким риском, совпадая с результатами в исходных данных. Поле отображения результатов также становится зеленой.



*Рис.28 – Прогнозирования соискатели с низким риском.*

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы поставленные задачи цели были достигнуты. Построена модель случайного леса с высокой точностью, оценена и проверена многими тестами. Успешно построил приложение с использованием модели случайного леса, делая прогнозы об уровне риска клиентов в процессе одобрения кредита.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Е. А. Соколов* Решающие деревья- 26 января 2018 г. P 3-6.
2. Машинное обучение: базовый алгоритм дерева решений [Электронный ресурс]. - Режим доступа: <https://russianblogs.com/article/1646812493/>
3. *C. П. Чистяков* Труды Карельского научного центра РАН / Случайные леса: обзор – № 1. 2013. С. 117–136.
4. *Siroky D.* Navigating Random Forests and related advances in algorithmic modeling - Statistics Surveys. 2009. Vol. 3. P. 147–163.
5. *Breiman L.* Out-of-bag estimation - Technical report, Statistics Department University of California, Berkeley. 1996. P. 1-13.
6. *Bylander T.* Estimating generalization error on twoclass datasets using outofbag estimates - Machine Learning. 2002. Vol. 48. P. 287–297.
7. Credit card classification – clean date [Электронный ресурс]. - Режим доступа: <https://www.kaggle.com/datasets/samuelcortinhas/credit-card-classification-clean-data>

# ПРИЛОЖЕНИЕ А

from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox

from lib.mainwindow import Ui\_MainWindow

import pickle

import sys

from RandomForest import RandomForest

import pandas as pd

class mywindow(QMainWindow):

def \_\_init\_\_(self):

super(mywindow, self).\_\_init\_\_()

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.ui.btn\_check.clicked.connect(self.onCheckBtnClick)

with open('./model/forest.model1', 'rb') as randomforestmodel:

self.my\_forest = pickle.load(randomforestmodel)

def mapping(self, var, value):

if var == 'Income\_type':

if value == 'Commercial associate':

return 0

if value == 'Pensioner':

return 1

if value == 'State servant':

return 2

if value == 'Student':

return 3

if value == 'Working':

return 4

if var == 'Education\_type':

if value == 'Academic degree':

return 0

if value == 'Higher education':

return 1

if value == 'Incomplete higher':

return 2

if value == 'Lower secondary':

return 3

if value == 'Secondary / secondary special':

return 4

if var == 'Family\_status':

if value == 'Civil marriage':

return 0

if value == 'Married':

return 1

if value == 'Separated':

return 2

Листинг 12. Приложение для прогнозирования риска.

if value == 'Single / not married':

return 3

if value == 'Widow':

return 4

if var == 'Housing\_type':

if value == 'Co-op apartment':

return 0

if value == 'House / apartment':

return 1

if value == 'Municipal apartment':

return 2

if value == 'Office apartment':

return 3

if value == 'Rented apartment':

return 4

if value == 'With parents':

return 5

if var == 'Occupation\_type':

if value == 'Accountants':

return 0

if value == 'Cleaning staff':

return 1

if value == 'Cooking staff':

return 2

if value == 'Core staff':

return 3

if value == 'Drivers':

return 4

if value == 'HR staff':

return 5

if value == 'High skill tech staff':

return 6

if value == 'IT staff':

return 7

if value == 'Laborers':

return 8

if value == 'Low-skill Laborers':

return 9

if value == 'Managers':

return 10

if value == 'Medicine staff':

return 11

if value == 'Other':

return 12

if value == 'Private service staff':

return 13

if value == 'Realty agents':

return 14

if value == 'Sales staff':

return 15

if value == 'Secretaries':

return 16

if value == 'Security staff':

return 17

if value == 'Waiters/barmen staff':

return 18

return None

def onCheckBtnClick(self):

try:

Gender = int(self.ui.gender.currentText())

Own\_car = int(self.ui.car.currentText())

Own\_property = int(self.ui.property.currentText())

Work\_phone = int(self.ui.workphone.currentText())

Phone = int(self.ui.phone.currentText())

Email = int(self.ui.work\_phone.currentText())

Years\_employed = float(self.ui.year\_work.toPlainText())

Num\_children = int(self.ui.number\_childrens.toPlainText())

Num\_family = int(self.ui.number\_family.toPlainText())

Account\_length = int(self.ui.number\_card.toPlainText())

Total\_income = float(self.ui.total\_income.toPlainText())

Age = float(self.ui.yearsold.toPlainText())

Unemployed = int(self.ui.unemployee.currentText())

Income\_type = str(self.ui.type\_income.currentText())

Education\_type = str(self.ui.type\_education.currentText())

Family\_status = str(self.ui.famil\_status.currentText())

Housing\_type = str(self.ui.house\_type.currentText())

Occupation\_type = str(self.ui.work\_type.currentText())

except Exception as e:

msgBox = QMessageBox()

msgBox.setText(str(e))

msgBox.show()

msgBox.exec()

return

row = {'Gender' :Gender, 'Own\_car' :Own\_car, 'Own\_property' :Own\_property, 'Work\_phone' :Work\_phone, 'Phone' :Phone, 'Email' :Email, 'Unemployed' :Unemployed,

'Num\_children' :Num\_children, 'Num\_family' :Num\_family, 'Account\_length' :Account\_length, 'Total\_income' :Total\_income, 'Age' :Age,

'Years\_employed' :Years\_employed, 'Income\_type' :self.mapping('Income\_type', Income\_type),

'Education\_type' :self.mapping('Education\_type', Education\_type), 'Family\_status' :self.mapping('Family\_status', Family\_status),

'Housing\_type' :self.mapping('Housing\_type', Housing\_type), 'Occupation\_type' :self.mapping('Occupation\_type', Occupation\_type)}

try:

df = pd.DataFrame(row, index=[0])

result = self.my\_forest.take\_prediction(df.iloc[0])

if result == 0:

self.ui.lineEdit\_result.setStyleSheet("background-color: rgb(155, 242, 145);")

self.ui.lineEdit\_result.setText('Низкий риск :)')

else:

self.ui.lineEdit\_result.setStyleSheet("background-color: rgb(232, 95, 104);")

self.ui.lineEdit\_result.setText('Высокий риск :(')

except Exception as e:

msgBox = QMessageBox()

msgBox.setText(str(e))

msgBox.show()

msgBox.exec()

return

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication([])

application = mywindow()

application.show()

sys.exit(app.exec())

# ПРИЛОЖЕНИЕ Б

Листинг 13. Случайный лес.

import time

from DecisionTree import train\_tree, prediction, entropy, gini

import pandas as pd

import numpy as np

import pickle

from sklearn.utils import shuffle

from sklearn import preprocessing

from collections import Counter

import seaborn as sn

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy\_score

from lib.auxiliary import oversample

class RandomForest:

def \_\_init\_\_(self, data, impurity\_func = entropy, max\_depth = 20,min\_samples\_split = 2, trees = 25):

self.data = data

self.y = str(data.columns[-1])

self.impurity\_func = impurity\_func

self.max\_depth = max\_depth

self.min\_samples\_split = min\_samples\_split

self.trees = trees

self.list\_tree = []

def create\_fores(self):

#Split data to any samples

data = self.data.copy()

list\_samples = []

size = (len(data) \* 80) // 100

for i in range(0, self.trees):

list\_samples.append(data.sample(size))

#create trees

t = 0

for df in list\_samples:

tree = train\_tree(df, self.y, self.impurity\_func, self.max\_depth, self.min\_samples\_split, 0)

t += 1

print('Tree {} was created'.format(t))

self.list\_tree.append(tree)

def test\_forest(self, df\_test):

list\_pred = []

total\_0 = (df\_test[str(df\_test.columns[-1])] == 0).sum()

total\_1 = (df\_test[str(df\_test.columns[-1])] == 1).sum()

print(total\_0, total\_1)

for i in range(len(df\_test)):

list\_result = []

for tree in self.list\_tree:

list\_result.append(int(prediction(tree, df\_test.iloc[i])))

counter = Counter(list\_result)

list\_pred.append(counter.most\_common(1)[0][0])

return list\_pred

def take\_prediction(self, df\_row):

list\_result = []

for tree in self.list\_tree:

list\_result.append(int(prediction(tree, df\_row)))

counter = Counter(list\_result)

return counter.most\_common(1)[0][0]

if \_\_name\_\_ == '\_\_main\_\_':

#Import data

df = pd.read\_table("./data/clean\_data.csv", delimiter=',')

df.drop('ID', axis = 1, inplace = True)

df = oversample(df)

df = shuffle(df)

le = preprocessing.LabelEncoder()

df["Income\_type"] = le.fit\_transform(df["Income\_type"])

df["Education\_type"] = le.fit\_transform(df["Education\_type"])

df["Family\_status"] = le.fit\_transform(df["Family\_status"])

df["Housing\_type"] = le.fit\_transform(df["Housing\_type"])

df["Occupation\_type"] = le.fit\_transform(df["Occupation\_type"])

df\_train = df[:13481]

df\_test = df[13481:]

#Train forest

randomforest = RandomForest(df\_train, entropy, 20, 2, 100)

t1 = time.time()

randomforest.create\_fores()

print("The random forest was created in {} seconds.".format(time.time() - t1))

with open('./model/forest.model1', 'wb') as randomforestmodel:

pickle.dump(randomforest, randomforestmodel)

#Test fores

with open('./model/forest.model1', 'rb') as randomforestmodel:

my\_forest = pickle.load(randomforestmodel)

t2 = time.time()

y\_pred = my\_forest.test\_forest(df\_test)

print("The testing was done in {} seconds.".format(time.time() - t2))

y\_test = df\_test['Y'].to\_numpy()

y\_pred = np.array(y\_pred).astype(int)

print("Accuracy: {}%".format(accuracy\_score(y\_test, y\_pred) \* 100))

confusion\_matrix = pd.crosstab(y\_test, y\_pred, rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion\_matrix, annot=True)

plt.show()

# ПРИЛОЖЕНИЕ В

Листинг 14. Древо решений.

import pandas as pd

import numpy as np

import pickle

from sklearn import preprocessing

from node import Node

from collections import Counter

from sklearn.utils import shuffle

from lib.auxiliary import oversample

def entropy(var):

try:

a = var.value\_counts() / var.shape[0]

entropy = np.sum(-a \* np.log2(a))

return(entropy)

except Exception as e:

print(e)

def information\_gain(target\_var, mask, func):

a = sum(mask)

b = mask.shape[0] - a

if (a == 0 or b ==0):

ig = 0

else:

ig = func(target\_var) - a / (a + b) \* func(target\_var[mask]) - b / (a + b) \* func(target\_var[-mask])

return ig

def max\_information\_gain\_split(var\_testing, target\_var, func):

split\_value = []

ig = []

# Create options

options = var\_testing.sort\_values().unique()

# Calculate ig for all values

for val in options:

mask = var\_testing <= val

val\_ig = information\_gain(target\_var=target\_var, mask = mask, func=func)

# Append results

ig.append(val\_ig)

split\_value.append(val)

if len(ig) == 0:

return None, None

else:

best\_ig = max(ig)

best\_ig\_index = ig.index(best\_ig)

best\_split = split\_value[best\_ig\_index]

return best\_ig, best\_split

def get\_best\_option(df, var\_target\_name, func):

variable = ''

max\_best\_ig = 0

split\_value = 0

for var in df.columns:

if str(var) != var\_target\_name:

best\_ig, best\_split = max\_information\_gain\_split(df[str(var)], df[var\_target\_name], func)

if (best\_ig == None):

return None, None, None

if best\_ig > max\_best\_ig:

max\_best\_ig = best\_ig

variable = str(var)

split\_value = best\_split

if max\_best\_ig != 0:

return variable, max\_best\_ig, split\_value

else:

return None, None, None

def make\_split(variable, value, data):

data\_1 = data[data[variable] <= value]

data\_2 = data[(data[variable] <= value) == False]

return data\_1,data\_2

def train\_tree(data, y, impurity\_func = entropy, max\_depth = 5,min\_samples\_split = 10, counter=0):

if len(data) >= min\_samples\_split and counter <= max\_depth:

variable, max\_best\_ig, split\_value = get\_best\_option(data, y, func = impurity\_func)

if max\_best\_ig is not None:

left, right = make\_split(variable, split\_value, data)

if left.empty or right.empty:

return Node(value=Counter(data[y]).most\_common(1)[0][0])

yes\_answer = train\_tree(left, y, impurity\_func, max\_depth, min\_samples\_split, counter + 1)

no\_answer = train\_tree(right, y, impurity\_func, max\_depth, min\_samples\_split, counter + 1)

return Node(

feature = variable,

threshold = split\_value,

data\_left = yes\_answer,

data\_right = no\_answer,

gain = max\_best\_ig)

return Node(value=Counter(data[y]).most\_common(1)[0][0])

def prediction(tree, row):

if tree.value != None:

return tree.value

if row[tree.feature] <= tree.threshold:

return prediction(tree.data\_left, row)

if row[tree.feature] > tree.threshold:

return prediction(tree.data\_right, row)

# ПРИЛОЖЕНИЕ Г

Листинг 15. Избыточная выборка.

import pandas as pd

def oversample(df):

classes = df.Y.value\_counts().to\_dict()

most\_value = max(classes.values())

classes\_list = []

for cl in classes:

classes\_list.append(df[df['Y'] == cl])

classes\_sample = []

for i in range(1,len(classes\_list)):

classes\_sample.append(classes\_list[i].sample(most\_value, replace = True))

df\_maybe = pd.concat(classes\_sample)

result = pd.concat([df\_maybe, classes\_list[0]], axis=0)

result = result.reset\_index(drop = True)

return result

# ПРИЛОЖЕНИЕ Д

Листинг 16. Узел дерева решений.

class Node:

def \_\_init\_\_(self, feature=None, threshold=None, data\_left=None, data\_right=None, gain=None, value=None):

self.feature = feature

self.threshold = threshold

self.data\_left = data\_left

self.data\_right = data\_right

self.gain = gain

self.value = value