**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение**

**Высшего профессионального образования**

**Московский технический университет связи и информатики**

Кафедра «Математическая кибернетика и информационные технологии»

**Отчет по курсовой работе**

**Решение задачи классификации с применением методов Data Mining**

по дисциплине «Data mining и базы данных»

Выполнил: Разумов И.В.

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

подпись дата

Проверил: Иевлев К.О.

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

подпись дата

**Москва 2023**

# **Оглавление**

[**Оглавление** 2](#_Toc134872597)

[**Цель работы** 4](#_Toc134872598)

[**Задание на курсовую работу** 5](#_Toc134872599)

[**Выполнение** 6](#_Toc134872600)

[**Глава 1 – выбор датасета и анализ данных** 6](#_Toc134872601)

[**Глава 2 – предобработка исходных данных, а также выбор и обучение моделей** 15](#_Toc134872602)

[**Глава 3 – анализ полученных результатов** 24](#_Toc134872603)

[**Выводы** 31](#_Toc134872604)

[**Список используемой литературы** 32](#_Toc134872605)

[**Приложение:** 33](#_Toc134872606)

**Введение**

Data mining, также известный как knowledge discovery in data (KDD), это процесс выявления закономерностей и другой ценной информации из больших наборов данных. С развитием технологий хранения данных и ростом big data, применение методов data mining быстро ускорилось за последние несколько десятилетий, помогая компаниям преобразовывать их необработанные данные в полезные знания.

Data mining использует методы на стыке машинного обучения, статистики и систем баз данных. Целью data mining является извлечение информации (с помощью интеллектуальных методов) из набора данных и преобразование информации в понятную структуру для дальнейшего использования.

Процесс data mining обычно состоит из четырех основных шагов: установление целей, сбор и подготовка данных, применение алгоритмов data mining и оценка результатов. Data mining используется для генерации описаний и прогнозов о целевом наборе данных. Данные описываются через наблюдения закономерностей, ассоциаций и корреляций. Также данные классифицируются и группируются с помощью методов классификации и регрессии, а также выявляются выбросы для использования в случаях, например, обнаружения спама.

Машинное обучение продолжает развиваться и расширять свои возможности. В настоящее время машинное обучение используется во многих областях, таких как медицина, финансы, реклама и маркетинг, безопасность и многих других.

Одной из сфер применения машинного обучения является медицина. В качестве темы курсовой работы была выбрана тема предсказывания наличие депрессии у человека, на основе его данных.

Депрессия является распространенным нарушением психического здоровья. Согласно оценкам, депрессией страдает 5% взрослого населения мира. В развитии депрессии участвует множество факторов.  Причины у каждого человека, страдающего депрессией, свои: низкая самооценка, потеря близкого человека, безденежье, послеродовой период у женщин и многое другое.

Несмотря на наличие проверенных и эффективных методов лечения психических расстройств, более 75% людей в странах с низким и средним уровнем дохода не получают никакого лечения. К факторам, препятствующим получению эффективного лечения, относятся недостаточные капиталовложения в охрану психического здоровья, нехватка подготовленных медицинских работников и социальная стигматизация людей, страдающих психическими расстройствами.

Одним из главных факторов правильного подхода к лечению – диагностика данного заболевания. Так как методы машинного обучения распространены в медицине, то можно использовать их и для диагностики депрессии. В качестве дальнейшего исследования будет взят датасет, содержащий

# **Цель работы**

Целью выполнения данной курсовой работы является закрепление и углубление теоретических знаний в области data mining. Также необходимо применить полученные теоретические знания на практике. Для этого необходимо практически реализовать инструменты Data Mining к набору данных, с целью выполнения следующих задач: предварительного анализа имеющихся данных, предобработки данных (удаление данных, не являющихся для дальнейших задач ценными), обучения методами машинного обучения, а также анализ полученных результатов.

В качестве исходных данных у нас имеется датасет, полученный в ходе исследования факторов, способных повлиять на депрессию людей, живущих в сельской местности. Задача модели ML --- научиться определять склонность к депрессии (0 --- не страдает, 1 --- страдает). Это типичная задача **бинарной классификации**

# **Задание на курсовую работу**

Для выполнения курсовой работы необходимо составить чёткий алгоритм выполнения курсовой работы. Данный алгоритм будет включать себя следующие пункты:

1) Изучение полученного набора данных с помощью различных статистических методов.

2) Подготовка данных к применению методов машинного обучения. Данный пункт содержит в себе следующие действия: избавление данных от пропусков, в случае их наличия; определение ценных атрибутов данных, а также определение не ценных атрибутов данных с последующим их удалением.

3) Выбор моделей машинного обучения, пригодных для решения поставленной задачи.

4) Обучение выбранных моделей машинного обучения на заранее подготовленных данных

5) Сравнение эффективность выбранных алгоритмов, сделать вывод о наилучшем.

# **Выполнение**

Далее выполнение будет поделено на 3 главы:

1. Первая глава будет посвящена выбору датасета, а также анализу его данных
2. Далее, 2 глава будет посвящена предобработке данных на основе изученных данных из первой главы, далее будут выбраны подходящие для решения нашей задчи методы машинного обучения, в заключении главы будет произведено обучение моделей.
3. Последняя глава будет посвящена анализу полученных данных из второй главы. С помощью различных метрик посмотрим качество обучения моделей. Также оценим скорость их обучения.

# **Глава 1 – выбор датасета и анализ данных**

1. Поиск необходимых данных

Для решение поставленной задачи, необходимо было найти датасет, содержащий в себе информацию о людях, страдающих или не страдающих депрессией. В результате, на платформе kaggle был найден датасет, удовлетворяющий указанным условиям.

2. Изучение предложенного описания датасета

Далее взглянем на предложенное описание датасета. Датасет содержит в себе следующие параметры:

* Survey\_id
* Ville\_id
* Sex
* Age
* Married
* Number\_children
* education\_level
* total\_members (in the family)
* gained\_asset -
* durable\_asset -
* save\_asset -
* living\_expenses
* other\_expenses
* incoming\_salary
* incoming\_own\_farm
* incoming\_business
* incoming\_no\_business
* incoming\_agricultural
* farm\_expenses
* labor\_primary
* lasting\_investment
* no\_lasting\_investmen
* depressed: [ 0: Нет депрессии] или [1: есть депрессия]

3. Изучение датасета

После этого начнём изучать данные. Далее все последующие действия будем выполнять в IDE. Сначала считаем данные с помощью библиотеки Pandas. С помощью этой же библиотеки посмотрим на первые 5 строк нашего датасета.

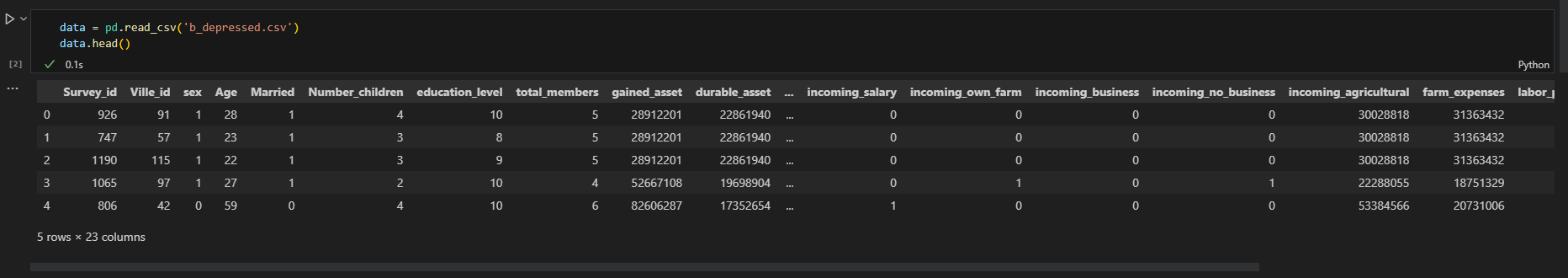


Рисунок 1 – Чтение датасета и вывод первых 5 элементов

Узнаем размер датасета с помощью команды data.shape:

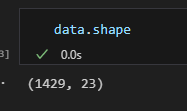


Рисунок 2 – Размер датасета

**Пропуски**

В любом датасете по тем или иным причинам могут содержаться пропущенные значения. Существует много стратегий избавления от пропущенных данных. Например, один из вариантов - это заполнение средним значением или медианой, либо преобразование категориальных переменных в числовые данные (one-hot encoding), а также удаление столбцов. Изучим количество пропусков в датасете. Для этого применим метод isna().sum()

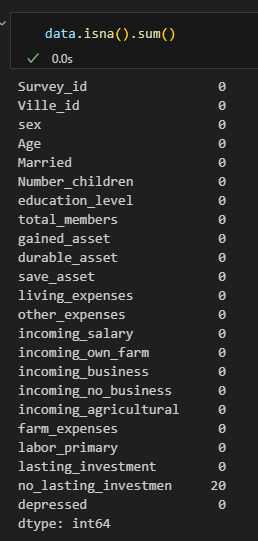


Рисунок 3 – Количество пропусков в датасете

Как мы видим, пропусков немного, всего 20 в колонке no\_lasting\_investmen. В данном случае пропуски можно заменить либо медианой, либо среднем арифметическим колонки.

Далее применим метод data.info(), чтобы узнать тип переменных каждой колонки.

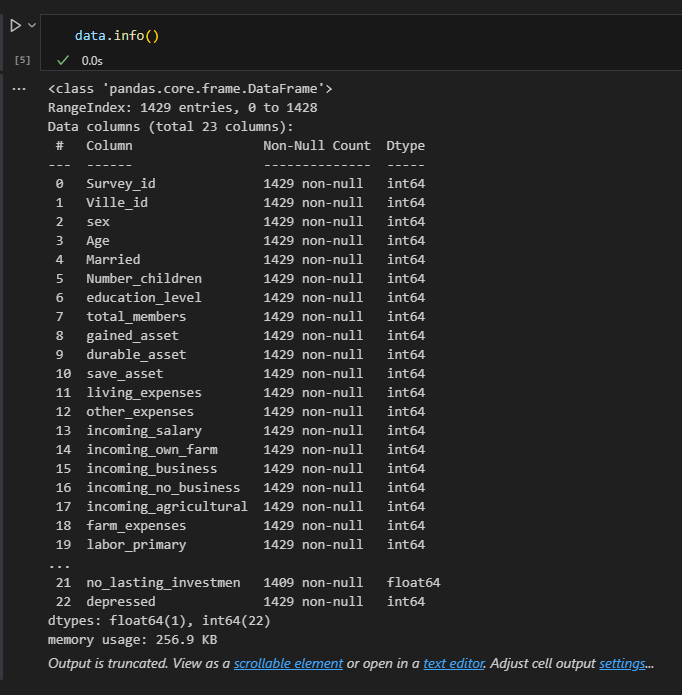


Рисунок 4 – Применение команды data.info(),

**Уникальные значения**

После этого изучим количество уникальных значений в каждом столбце. Для этого применим команду data.nunique(),

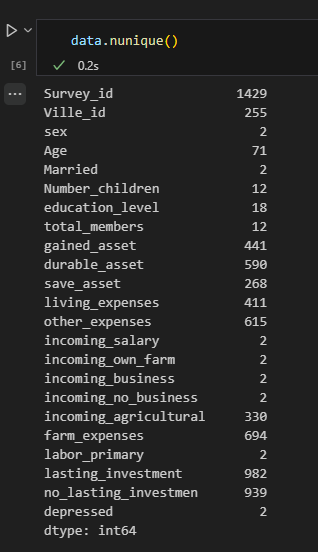


Рисунок 5 – Количество уникальных значений в каждой колонке

**Матрица корреляции**

Далее, воспользуемся матрицей корреляции. Матрица корреляции - это квадратная таблица, которая представляет корреляцию пар всех переменных. Она используется для подведения итогов большого количества данных, где цель состоит в том, чтобы увидеть закономерности. В целом, матрица корреляции является удобным инструментом для обобщения набора данных и выявления закономерностей между переменными.

Чтобы произвести количественную оценку взаимосвязи между двумя переменными, обычно использует коэффициент корреляции Пирсона, который является мерой линейной связи между двумя переменными. Он может принимать значения от -1 до 1.

* -1 указывает на совершенно отрицательную линейную корреляцию между двумя переменными;
* 0 указывает на отсутствие линейной корреляции между двумя переменными;
* 1 указывает на совершенно положительную линейную корреляцию между двумя переменными.

Коэффициент корреляции Пирсона r рассчитывается формулой:

где Mx - среднее арифметическое от x;

My - среднее арифметическое для y;

x и y - две переменных, для которых вычисляется корреляция.

Теперь построим матрицу корреляции для наших данных.

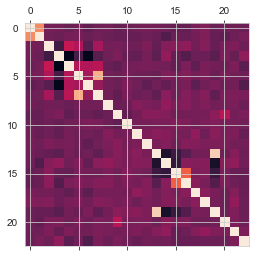


Рисунок 6 – Матрица корреляции

Для удобства немного изменим отображение матрицы: добавим подписи и шкалу значений.

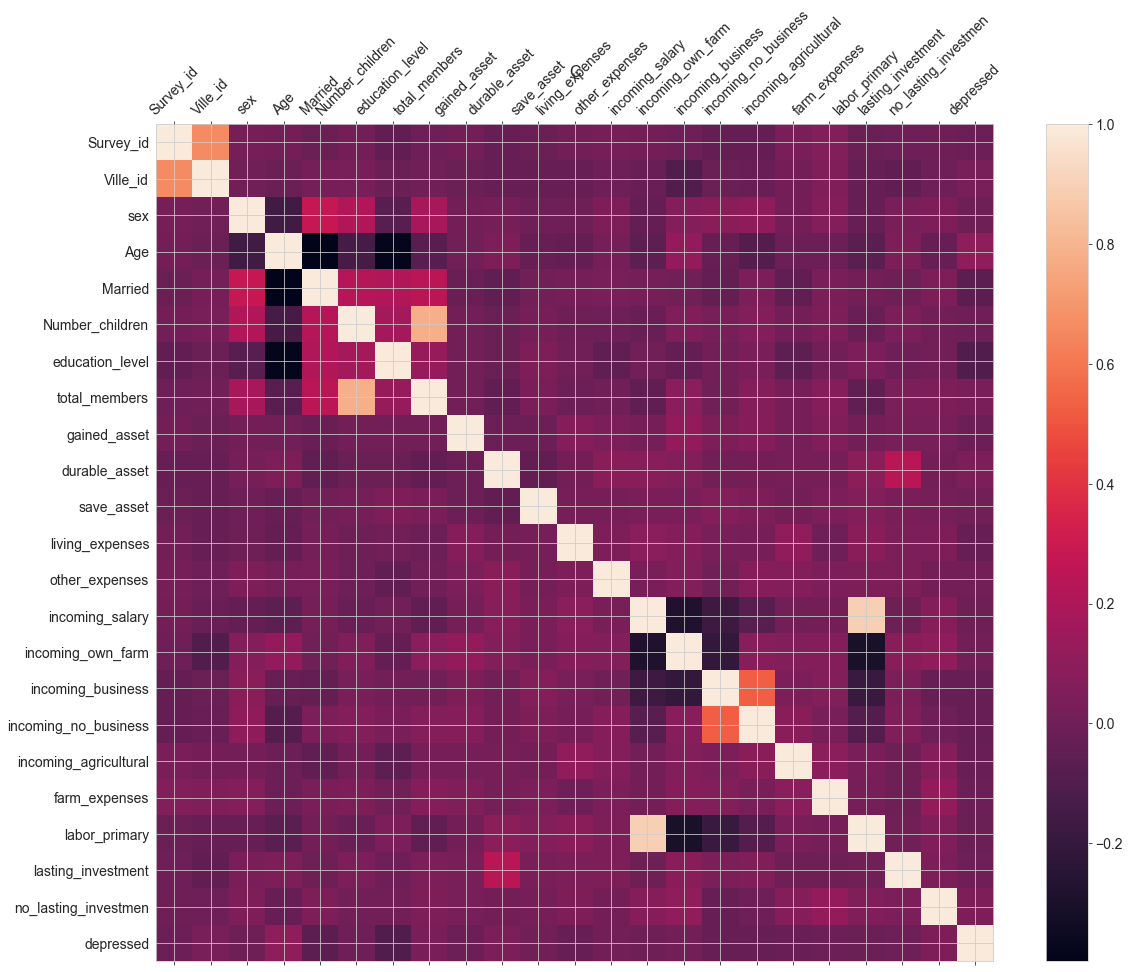
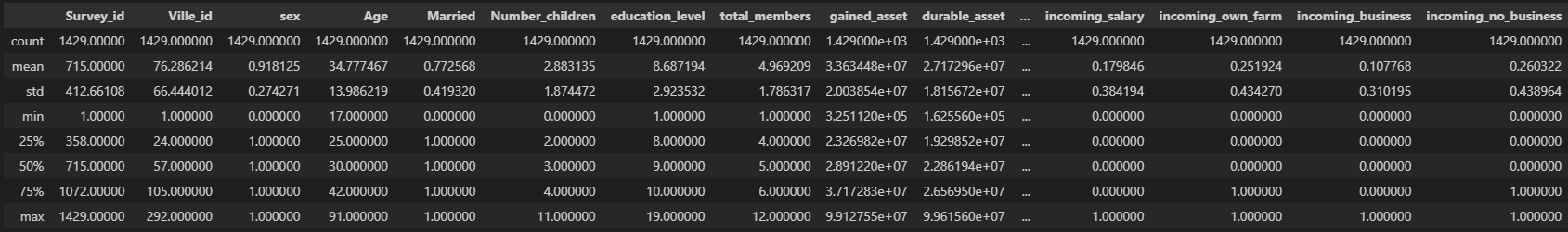


Рисунок 7 – Улучшенная матрица корреляции

Из данной матрицы можно сделать вывод, что некоторые параметры можно удалить, т.к. они имеют высокий показатель корреляции. Запомним их название, чтобы не забыть их удалить, когда будем

Далее, воспользуемся методом describe, чтобы узнать некоторые данные по каждому столбцу.





Рисунки 8 и 9 – Применение метода describe

**Визуализация данных**

Одним из вариантов визуализации данных является гистограмма распределения.

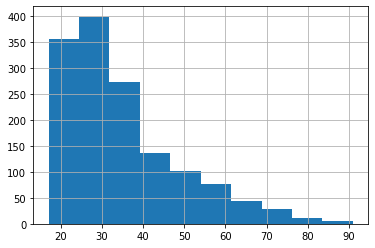


Рисунок 10 – Гистограмма по «Age»

Также, с помощью функции pairplot библиотеки seaborn можно увидеть распределение между несколькими числовыми переменными. Данная функция используется для создания матрицы диаграмм рассеяния для визуализации попарных отношений между переменными в наборе данных.

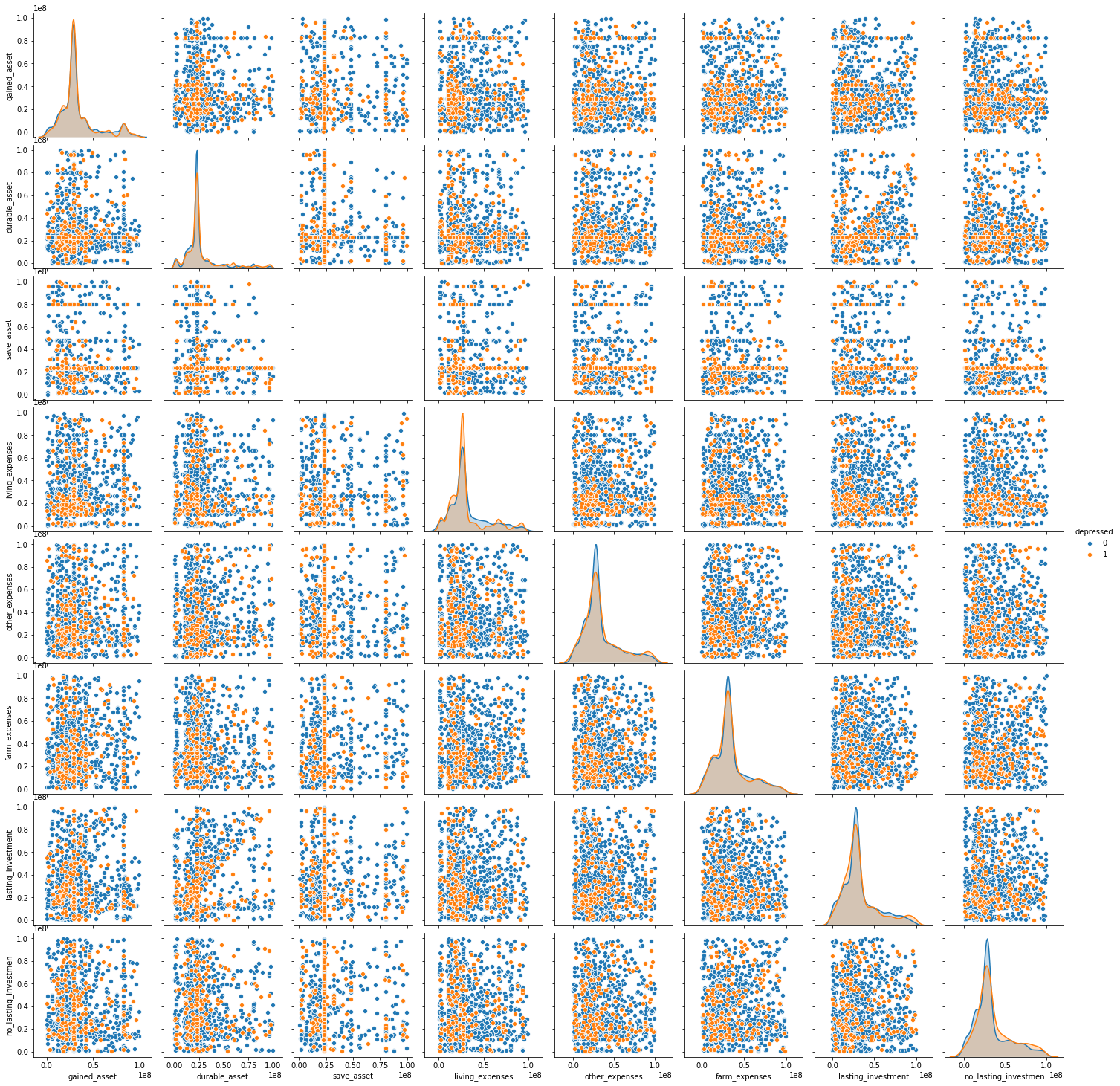


Рисунок 11 – Визуализация распределения между несколькими числовыми переменными с помощью функции pairplot библиотеки seaborn

# **Глава 2 – предобработка исходных данных, а также выбор и обучение моделей**

**Предобработка данных**

После изучения датасета, необходимо произвести предобработку данных для дальнейшего обучения, для этого удалим ненужные колонки и заполним возникшие пропуски, в качестве значение для замены возьмём медиану колонки. Также удалим сильно коррелирующие данные. На рисунке ниже представлен результат предобработки данных:

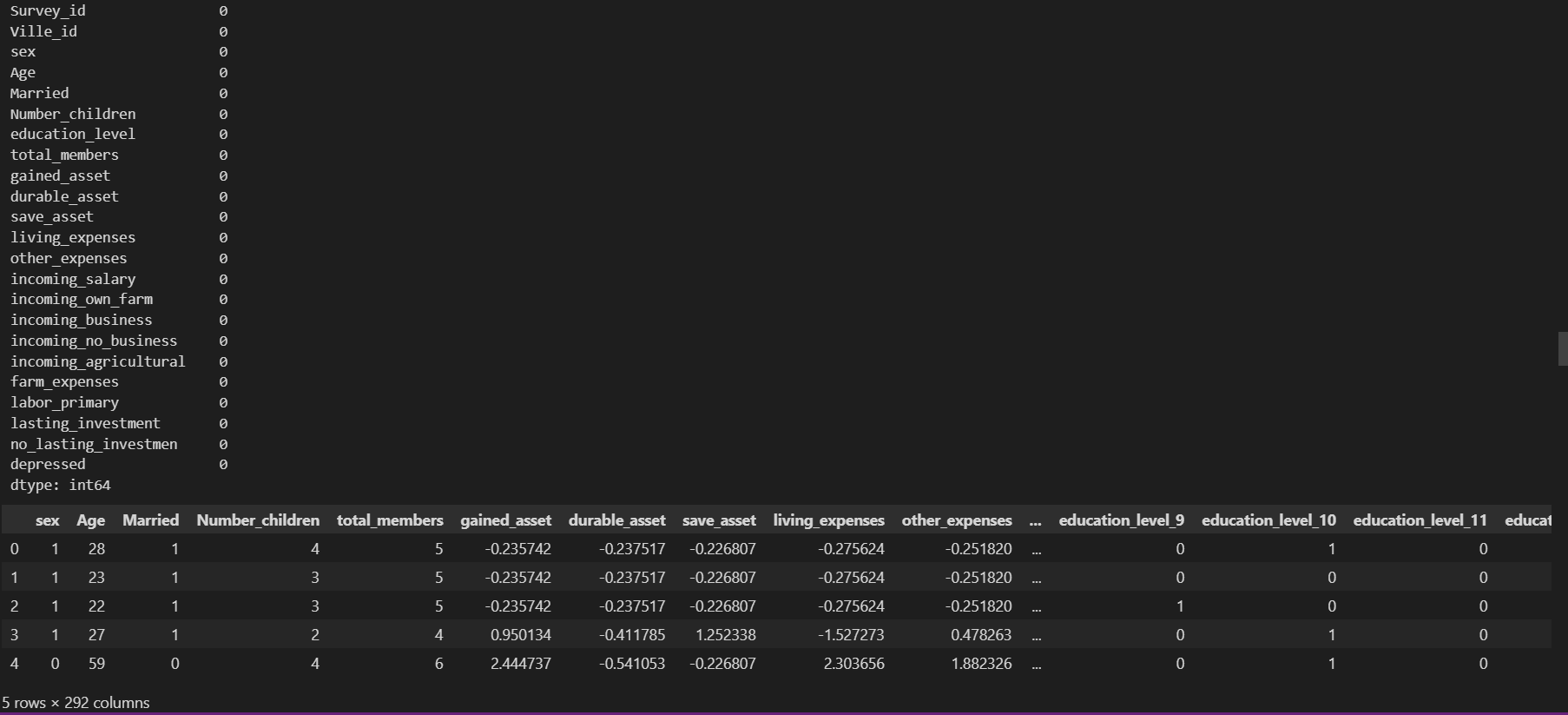


Рисунок 12 – Визуализация распределения между несколькими

**Выбор моделей и обучение**

Теперь, после предварительной предобработки данных необходимо выбрать модели машинного обучения, которые подойдут для решения поставленной задачи классификации. Для этого могут подойти следующие алгоритмы машинного обучения:

* Метод опорных векторов (Support Vector Machines)
* Классификатор дерева решений (Decision Tree Classifier)
* Случайный лес (Random Forests)
* Наивный байесовский метод (Naive Bayes)
* Логистическая регрессия

Рассмотрим каждый алгоритм более подробно.

**Логистическая регрессия**

Логистическая регрессия представляет собой одну из моделей, которые относятся к линейному классификатору. Простыми словами, задачей линейного классификатора является предсказание целевых значений  от переменных (регрессоров) . При этом считается, что зависимость между признаками  и целевыми значениями  линейная. Отсюда собственно и название классификатора — линейный. Если очень грубо обобщить, то в основе модели логистической регрессии лежит предположение о наличии линейной зависимости между признаками  и целевыми значениями . Вот она — связь.

Логистическая регрессия или логит-регрессия – это статистическая модель, применяемая для предсказания вероятности возникновения некоторого события по значениям множества переменных, как количественных, так и категориальных.

**Простота**

Модели логистической регрессии математически менее сложны, чем другие методы машинного обучения. Таким образом, вы можете внедрить их, даже если никто из вашей команды не обладает глубоким опытом в области машинного обучения.

**Скорость**

Модели логистической регрессии могут обрабатывать большие объемы данных с высокой скоростью, поскольку они требуют меньших вычислительных мощностей, таких как память и способность обработки. Это делает их идеальными для организаций, которые начинают с проектов машинного обучения, чтобы быстро добиться успеха.

**Гибкость**

Логистическую регрессию можно использовать для поиска ответов на вопросы, которые имеют два или более конечных результатов. Ее также можно использовать для предварительной обработки данных. Например, с помощью логистической регрессии можно сортировать данные с большим диапазоном значений, например, банковские транзакции, в меньший конечный диапазон значений. Затем можно обработать этот небольшой набор данных, используя другие методы машинного обучения для более точного анализа.

**Наглядность**

Логистический регрессионный анализ позволяет разработчикам лучше понять внутренние программные процессы, чем другие методы анализа данных. Устранение неполадок и исправление ошибок также упрощаются, поскольку вычисления менее сложны.

Цель логистической регрессии такая же, как и у линейной: смоделировать yi как линейную функцию от xi, однако для бинарных значений yi это не так просто. Линейная комбинация признаков, такая как wxi+b, — это функция, простирающаяся от минус бесконечности до плюс бесконечности, тогда как yi имеет только два возможных значения. Если определить отрицательную метку как 0, а положительную метку как 1, достаточно найти простую непрерывную функцию с областью значений (0, 1). В этом случае, если значение, возвращаемое моделью для образца x, ближе к 0, ему присваивается отрицательная метка; иначе образец маркируется как положительный. Одной из функций, обладающих таким свойством, является стандартная логистическая функция (также известная как логистический сигмоид):

Функция модели логистической регрессии:

Взглянув на график стандартной логистической функции, можно заметить, насколько хорошо она соответствует нашей цели классификации: если соответствующим образом оптимизировать значения w и b, результат f(x) можно интерпретировать как вероятность, что yi будет иметь положительное значение.

Например, если она выше или равна пороговому значению 0.5, мы бы сказали, что класс x положителен; иначе — отрицателен.

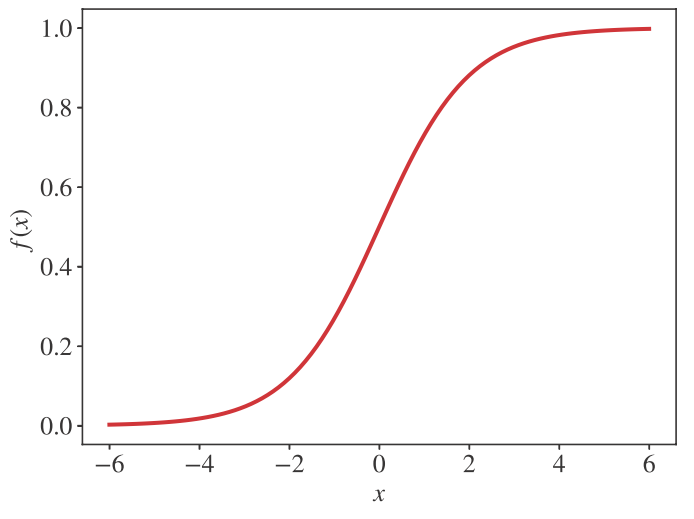


Рисунок 13 – График логистической регрессии

**Наивный Байес**

Наивный Байес — простой, но удивительно эффективный алгоритм.

Модель состоит из двух типов вероятностей, которые рассчитываются с помощью тренировочных данных:

1. Вероятность каждого класса.
2. Условная вероятность для каждого класса при каждом значении x.

После расчёта вероятностной модели её можно использовать для предсказания с новыми данными при помощи теоремы Байеса. Если у вас вещественные данные, то, предполагая нормальное распределение, рассчитать эти вероятности не составляет особой сложности.

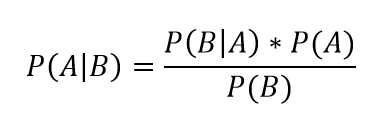


Рисунок 14 – Теорема Байеса

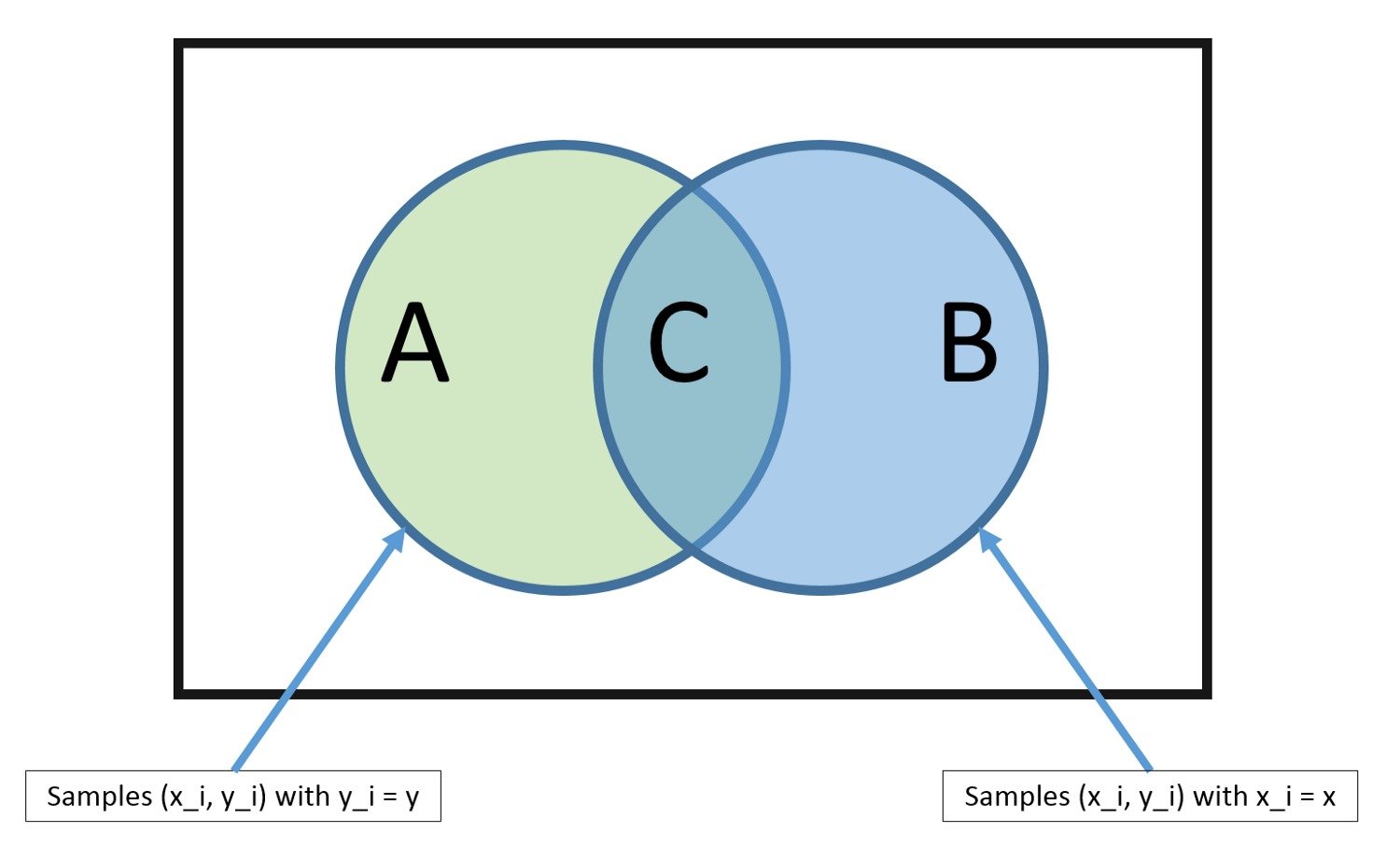


Рисунок 15 – Графическое объяснение Наивного Байеса

Наивный Байес называется наивным, потому что алгоритм предполагает, что каждая входная переменная независимая. Это сильное предположение, которое не соответствует реальным данным. Тем не менее данный алгоритм весьма эффективен для целого ряда сложных задач вроде классификации спама или распознавания рукописных цифр.

**Решающие деревья**

Решающие деревья - это метод машинного обучения, который используется для решения задач классификации и регрессии. Они представляют собой древовидную структуру, в которой каждый узел представляет собой тест на один из признаков объекта. В зависимости от результата теста мы переходим к одному из дочерних узлов и повторяем процесс до тех пор, пока не достигнем листового узла, который содержит ответ на задачу.

Построение решающего дерева начинается с корневого узла, который содержит все объекты обучающей выборки. Далее происходит разбиение выборки на две части в зависимости от значения одного из признаков. Этот процесс повторяется для каждого дочернего узла до тех пор, пока не будет достигнут критерий остановки.

Решающие деревья имеют ряд преимуществ перед другими методами машинного обучения. Они легко интерпретируются и могут быть использованы для классификации и регрессии как на числовых, так и на категориальных данных. Кроме того, они могут работать с пропущенными значениями и выбросами.

Решающие деревья могут использоваться в различных областях, таких как финансы, медицина, бизнес и т.д.  Например, в финансовой сфере решающие деревья могут использоваться для принятия решений о выдаче кредитов. В медицине они могут использоваться для диагностики заболеваний. В бизнесе решающие деревья могут использоваться для прогнозирования продаж.

Модули для построения и исследования деревьев решений входят в состав множества аналитических платформ. Это удобный инструмент, применяемый в системах поддержки принятия решений и интеллектуального анализа данных.

**Дерево решений** задаётся алгоритмом классификации a(x) задающийся деревом (связным ациклическим графом) и имеющим следующий вид:

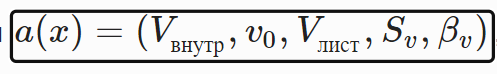


Рисунок 16 – Формула классификатора a(x)

Рассмотрим его более подробно:

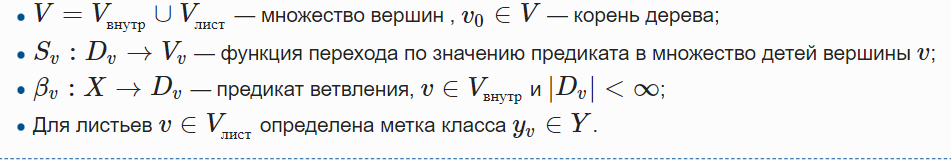


Рисунок 17 – Содержимое классификатора a(x)

**RandomForest**

Random Forest - это мета-оценщик, который сочетает вывод нескольких деревьев решений на различных подвыборках набора данных и использует усреднение для улучшения предсказательной точности и контроля над переобучением.

Random Forest является расширением метода бэггинга, так как он использует как бэггинг, так и случайность признаков для создания некоррелированного леса деревьев решений. Случайность признаков, также известная как бэггинг признаков или «метод случайного подпространства», генерирует случайное подмножество признаков, что обеспечивает низкую корреляцию между деревьями решений.

Random Forest является популярным алгоритмом машинного обучения благодаря его легкости использования и гибкости. Он обрабатывает как задачи классификации, так и задачи регрессии.

По сравнению с другими методами машинного обучения, теоретическая часть алгоритма **Random Forest** проста. У нас нет большого объема теории, необходима только формула итогового классификатора a(x):



Рисунок 18 – Формула итогового классификатора a(x)

Где

* **N** – количество деревьев;
* **i** – счетчик для деревьев;
* **b** – решающее дерево;
* **x** – сгенерированная нами на основе данных выборка.

*Стоит также отметить, что для задачи классификации мы выбираем решение голосованием по большинству, а в задаче регрессии – средним.*

**SVM**

Метод опорных вектором (SVM) требует, чтобы положительная метка имела числовое значение +1, а отрицательная метка имела значения -1.

SVM рассматривает каждый вектор признаков как точку в многомерном пространстве (в данном случае пространство имеет n измерений). Алгоритм помещает все векторы признаков на воображаемый n-мерный график и рисует воображаемую (n-1)-мерную линию (гиперплоскость), которая отделяет данные с положительными метками от данных с отрицательными метками. Граница, разделяющая данные разных классов, в машинном обучении называется границей принятия решения.

Уравнение гиперплоскости задается двумя параметрами: вещественным вектором w той же размерности, что и входной вектор признаков x, и действительным числом b, например:

wx - b =0,

где выражение wx означает w (1) x (1) + w (2) x (2) +…+ w (D) x (D), а D - число измерений в векторе признаков x.

Прогнозируемую метку для некоторого входного вектора признаков x

можно выразить так:

y = sign(wx – b),

где sign — это математический оператор, принимающий произвольное значение и возвращающий +1, если входное значение является положительным числом, и –1, если входное значение является отрицательным числом.

Цель алгоритма обучения, в данном случае SVM, используя набор данных, найти оптимальные значения w\* и b\* для параметров w и b. После того как алгоритм обучения найдет эти оптимальные значения, модель f(x) будет определяться как:

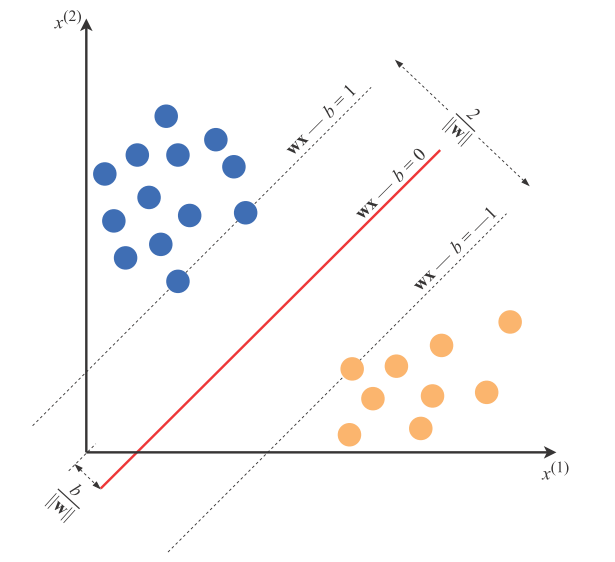


Рисунок 19 – График модели SVM

Теперь после того, как мы выбрали модели машинного обучения, которые по нашим оценкам хорошо покажут себя в решении нашей задачи бинарной классификации, необходимо провести их обучение. Применим описанные методы к изучаемому датасету. Для этого импортируем нужные модули, создадим и обучим каждую модель. Рассмотрим обучение модели на примере Наивного Байеса:

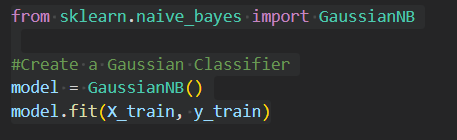


Рисунок 20 – Обучение модели Наивного Байеса

Далее, похожим образом обучим все модели.

# **Глава 3 – анализ полученных результатов**

Теперь проанализируем полученные результаты. Для этого воспользуемся различными метриками, показывающими, насколько хорошо наши модели смогли обучиться. Также, сравним время обучения каждого алгоритма

**Точность, полнота и F-мера**

Рассмотрим каждую метрику более подробно:

Accuracy - это метрика, используемая для оценки качества классификационных моделей. Она определяется как доля правильно классифицированных примеров от общего числа примеров.

Формула для вычисления accuracy выглядит следующим образом:

где TP - истинно-положительные (true positives), TN - истинно-отрицательные (true negatives), FP - ложно-положительные (false positives) и FN - ложно-отрицательные (false negatives).

* F1-score

F1-score - это метрика, используемая для оценки качества бинарных классификаторов. Она является гармоническим средним между точностью (precision) и полнотой (recall).

Формула для вычисления F1-score выглядит следующим образом:

где ​ и recall=​, TP - истинно-положительные (true positives), TN - истинно-отрицательные (true negatives), FP - ложно-положительные (false positives) и FN - ложно-отрицательные (false negatives).

* Precision

Precision (точность) - это метрика, используемая для оценки качества бинарных классификаторов. Она определяется как доля истинно-положительных результатов среди всех положительных результатов.

Формула для вычисления precision выглядит следующим образом:

* Recall

Recall (полнота) - это метрика, используемая для оценки качества бинарных классификаторов. Она определяется как доля истинно-положительных результатов среди всех положительных примеров.

Формула для вычисления recall выглядит следующим образом:

На основе матрицы ошибок можно построить следующие три метрики, каждая из которых лучше отображает результаты классификации для несбалансированных классов. Это **точность** (precision), **полнота** (recall) и **F-мера** (f1\_score).

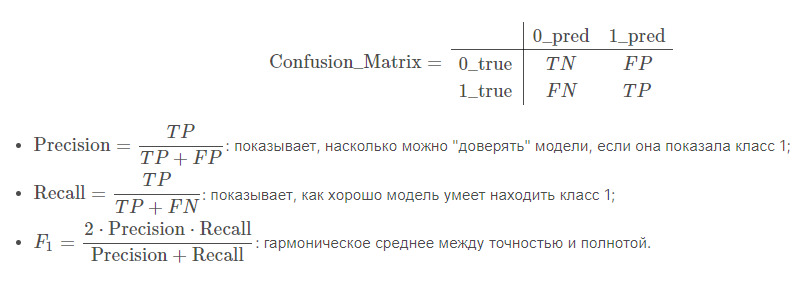


Рисунок 20– Confusion\_Matrix, **точность** (precision), **полнота** (recall) и **F-мера** (f1\_score)

Таким образом,

* если мы хотим минимизировать FP, то лучше в качестве метрики брать precision;
* если мы хотим минимизировать FN, то лучше в качестве метрики брать recall;
* если мы сомневаемся или хотим минимизировать оба значения, лучше подойдёт f1\_score.

Применим описанные методы к обученным моделям**.** На следующем рисунке представлены **точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для Логистической регрессии:

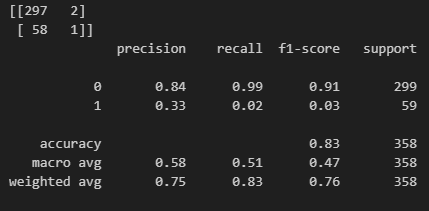


Рисунок 21 – **Точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для Логистической регрессии

На следующем рисунке представлены **точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для Наивного Байеса:

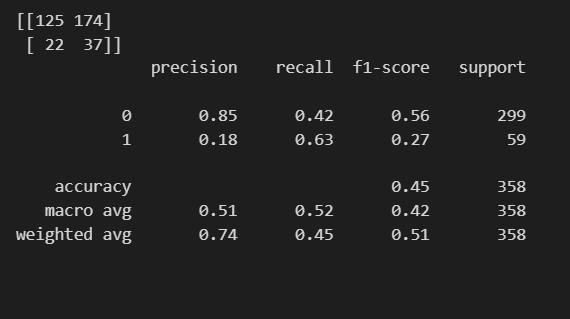


Рисунок 22 – **Точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для Наивного Байеса

На следующем рисунке представлены **точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для Решающего дерева:

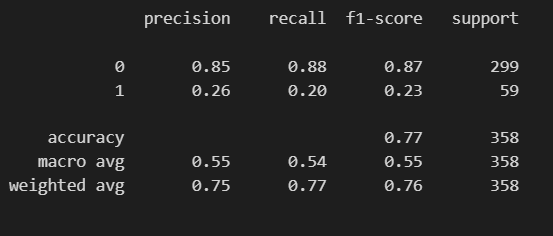


Рисунок 23 – **Точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для Решающего дерева:

На следующем рисунке представлены **точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для RandomForest:

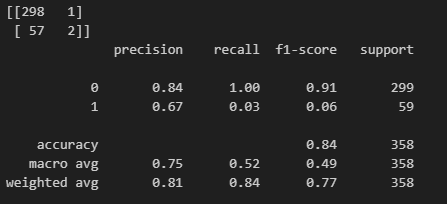


Рисунок **24** – **Точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для RandomForest

На следующем рисунке представлены **точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для SVM:

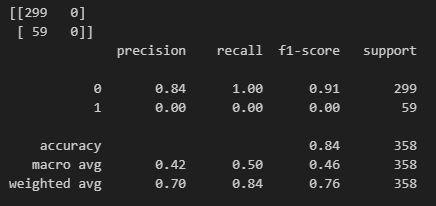


Рисунок 25 - **Точность** (precision), **полнота** (recall) и **F-мера** (f1\_score) для SVM

Ниже приведена таблица для значения Accuracy для каждого алгоритма:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Название алгоритма | | | | |
| LR | NB | DT | RF | SVM |
| Значение Accuracy | 0.83 | 0.45 | 0.77 | 0.83 | 0.83 |

Таблица 1 - Значения Accuracy для каждого алгоритма

Наконец, с помощью графика сравним эффективность каждого алгоритма и выбирим тот, который покажет самый лучший результат с тестовыми данными. Для этого воспользуемся библиотекой seaborn.

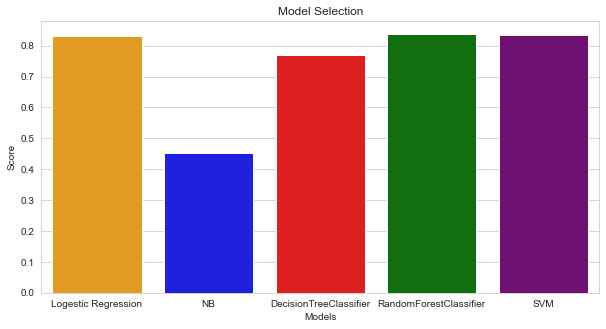
****

Рисунок 26 – График сравнения эффективности каждого алгоритма

Исходя из всех показателей можно сделать вывод, что самым эффективным алгоритмом является RandomForest.

**Скорость обучения.**

Одним из важнейших показателей качества обучения является скорость обучения. В таблице ниже сравним скорость выполнения каждого алгоритма:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Название алгоритма | | | | |
| LR | NB | DT | RF | SVM |
| Скорость обучения | 0,1 c | 0,01c | 1,4 c | 0,4 | 0,37 c |

Таблица 2 – Скорость обучения для каждого алгоритма

Согласно таблице выше, самым быстрым по скорости вычисления алгоритмом является Наивный Байес.

Таким образом, мы успешно провели анализ полученных данных, провели предобработку, выбрали и применили алгоритмы машинного обучения, а также проанализировали качество обученных моделей.

# **Выводы**

В результатет данной работы, благодаря использованию моделей машинного обучения мы смогли предсказать наличие депрессии у человека по обучающим переменным путем подбора параметров для более оптимального процесса обучения. В связи с присутствием пропусков они были заменены на медиану той колонки, в которой они находились.

Чтобы понять, какие параметры наиболее значимые, была построена матрица корреляции. Также был произведен замер метрик эффективности у каждой модели.

По итогу было выяснено, что самыми лучшими моделями являются RandomForest - имеет лучшие показатели метрик; и Наивный Байес – имеет наименьшее время обучения.

# **Список используемой литературы**

1. Андрей Бурков - МАШИННОЕ ОБУЧЕНИЕ без лишних слов
2. Федотов Станислав, Синицин - Филипп Учебник по машинному обучению
3. Обзор самых популярных алгоритмов машинного обучения (tproger.ru)
4. Alex Mazanski - Машинное обучение для начинающих: алгоритм случайного леса (Random Forest)

# **Приложение:**

Далее представлен листинг программы:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read\_csv('b\_depressed.csv')

data.head()

data.shape

data.isna().sum()

data.info()

data.nunique()

plt.matshow(data.corr())

plt.show()

f = plt.figure(figsize=(19, 15))

plt.matshow(data.corr(), fignum=f.number)

plt.xticks(range(data.shape[1]), data.columns, fontsize=14, rotation=45)

plt.yticks(range(data.shape[1]), data.columns, fontsize=14)

cb = plt.colorbar()

cb.ax.tick\_params(labelsize=14)

plt.title('C',fontsize=16)

data.describe()

variables = ["gained\_asset", "durable\_asset", "save\_asset", "living\_expenses", "other\_expenses", "farm\_expenses", "lasting\_investment", "no\_lasting\_investmen","depressed"]

sns.pairplot(data[variables],hue='depressed')

hist = data['Age'].hist()

data['no\_lasting\_investmen'].fillna(data['no\_lasting\_investmen'].median(), inplace=True)

print(data.isna().sum())

df\_1 = data

# Дропнем ненужные столбцы

df\_2 = df\_1.drop(['Survey\_id', 'depressed'], axis=1)

# Переведём признаки "Номер виллы" и "Уровень образования" в бинарные

# \* мы не уверены на 100 %, что уровень образования ранговый, поэтому считаем его категориальным

df\_3 = pd.get\_dummies(df\_2, columns=['Ville\_id', 'education\_level'])

# Масштабирование

col\_names = data.columns.values # это имена всех столбцов

large\_numbers = [col for col in col\_names if data[col].mean() > 10000] # имена тех, у кого среднее > 10000

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_st = scaler.fit\_transform(df\_3[large\_numbers])

# Переприсвоим старым колонкам новые

df\_3[large\_numbers] = X\_st

df\_3.head()

from sklearn.linear\_model import LogisticRegression

X = df\_3

y = df\_1['depressed']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X, y, test\_size=0.25, random\_state=0)

log\_reg = LogisticRegression()

log\_reg.fit(X\_train, y\_train)

y\_pred = log\_reg.predict(X\_valid)

from sklearn.metrics import accuracy\_score

print(accuracy\_score(y\_valid, y\_pred))

from sklearn.metrics import precision\_score, recall\_score, confusion\_matrix, f1\_score, classification\_report

from sklearn.utils.multiclass import unique\_labels

cm = confusion\_matrix(y\_valid, y\_pred)

print(cm)

# Print the precision and recall, among other metrics

print(classification\_report(y\_valid, y\_pred, digits=2))

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

rf = RandomForestClassifier()

rf.fit(X\_train,y\_train)

y\_pred = rf.predict(X\_test)

print(accuracy\_score(y\_test, y\_pred))

cm = confusion\_matrix(y\_valid, y\_pred)

print(cm)

# Print the precision and recall, among other metrics

print(classification\_report(y\_valid, y\_pred, digits=2))

acc\_test = rf.score(X\_test , y\_test)

from sklearn.tree import DecisionTreeClassifier

depth\_num= range(50, 71, 2)

training\_acc= []

testing\_acc = []

for depth in depth\_num:

    tree\_model = DecisionTreeClassifier(max\_depth=depth,random\_state=42)

    tree\_model.fit(X\_train,y\_train)

    training\_acc.append(tree\_model.score(X\_train,y\_train))

    testing\_acc.append(tree\_model.score(X\_test,y\_test))

print("Training Accuracy Scores:", training\_acc[:3])

print("testing Accuracy Scores:", testing\_acc[:3])

y\_pred\_DT = tree\_model.predict(X\_test)

print(classification\_report(y\_test,y\_pred\_DT  , digits=2))

X\_train\_bayes, X\_test\_bayes, y\_train\_bayes, y\_test\_bayes = train\_test\_split(X, y,

                                                    test\_size=.27, )

from sklearn.naive\_bayes import GaussianNB

#Create a Gaussian Classifier

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_pred\_bayes = model.predict(X\_test)

print(accuracy\_score(y\_test, y\_pred\_bayes))

cm = confusion\_matrix(y\_test, y\_pred\_bayes)

print(cm)

# Print the precision and recall, among other metrics

print(classification\_report(y\_test, y\_pred\_bayes, digits=2))

from sklearn.svm import SVC

svc\_model= SVC()

svc\_model.fit(X\_train,y\_train)

y\_pred\_svc = svc\_model.predict(X\_test)

print(f"Training accuracy is {accuracy\_score(y\_valid, y\_pred\_svc)}")

cm = confusion\_matrix(y\_test, y\_pred\_svc)

print(cm)

# Print the precision and recall, among other metrics

print(classification\_report(y\_test, y\_pred\_svc, digits=2))

models = pd.DataFrame({

    "Models": ["Logestic Regression" , "NB", "DecisionTreeClassifier","RandomForestClassifier", "SVM"],

    "Score":[log\_reg.score(X\_test,y\_test) ,model.score(X\_test,y\_test) , tree\_model.score(X\_test,y\_test),acc\_test, svc\_model.score(X\_test,y\_test)]

})

models.sort\_values(by="Score" , ascending=False)

colors= ['orange' , 'blue','red','green','purple']

sns.set\_style('whitegrid')

plt.figure(figsize=(10,5))

sns.barplot(x=models['Models'],y=models['Score'], palette=colors )

plt.xlabel("Models")

plt.ylabel("Score")

plt.title("Model Selection")

plt.show()