МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГУ»)

Факультет прикладной математики, информатики и механики

Кафедра программного обеспечения и администрирования информационных систем

**Тема курсовой работы**

Курсовая работа

По дисциплине Машинное обучение

Направление 02.03.03 Математическое обеспечение и администрирование информационных систем

Профиль Проектирование и разработка информационных систем

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_ |  | И.О. Фамилия |
| Проверил | \_\_\_\_\_\_\_\_\_\_ | д.т.н., проф. | И.Л. Каширина |

Воронеж 2023

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc105272267)

[1 Использованные методы машинного обучения 4](#_Toc105272268)

[1.1 Метод логистической регрессии 4](#_Toc105272269)

[1.2 Метод дерева решений 10](#_Toc105272270)

[1.3 Метод случайного леса 14](#_Toc105272271)

[2 Решение задачи предсказания взятия клиентом кредита 17](#_Toc105272272)

[2.1 Постановка задачи 17](#_Toc105272274)

[2.2 Проведение предобработки данных поставленной задачи 19](#_Toc105272275)

[2.3 Обучение модели логистической регрессии 23](#_Toc105272276)

[2.4 Обучение модели случайного леса 24](#_Toc105272277)

[2.5 Обучение модели дерева решений 25](#_Toc105272278)

[ЗАКЛЮЧЕНИЕ 27](#_Toc105272280)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc105272281)

# ВВЕДЕНИЕ

Технология машинного обучения на основе анализа данных берёт начало в 1950 году, когда начали разрабатывать первые программы для игры в шашки. За прошедшие десятилетий общий принцип не изменился. Зато благодаря взрывному росту вычислительных мощностей компьютеров многократно усложнились закономерности и прогнозы, создаваемые ими, и расширился круг проблем и задач, решаемых с использованием машинного обучения.

Благодаря машинному обучению компьютеры учатся распознавать на фотографиях и рисунках не только лица, но и пейзажи, предметы, текст и цифры. Функция проверки грамматики сейчас присутствует в любом текстовом редакторе и даже в телефонах. Причем учитывается не только написание слов, но и контекст, оттенки смысла и другие тонкие лингвистические аспекты.

Главной целью курсовой работы является реализация таких алгоритмов машинного обучения, как логистическая регрессия, случайный лес и дерево решений для решения задачи банковского маркетинга.

В рамках данной курсовой работы необходимо выполнить задачи:

1. Применить метод логистической регрессии для поставленной задачи;
2. Применить метод случайного леса для поставленной задачи;
3. Применить метод дерева решений для поставленной задачи;
4. Сравнить точность данных методов при решении задачи банковского маркетинга.

# МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ

## Метод логистической регрессии

Логистическая регрессия – это способ определения зависимости между переменными. Для этого применяется логистическая функция (аккумулятивное логистическое распределение). Метод логистической регрессии востребован в следующих задачах:

* кредитный скоринг;
* замеры успешности проводимых рекламных кампаний;
* прогноз прибыли с продажи определённого товара;
* оценка вероятности землетрясения в конкретную дату.

Все регрессионные модели могут быть записаны в виде формулы 1:

|  |  |
| --- | --- |
|  | (1) |

В множественной линейной регрессии предполагается, что зависимая переменная является линейной функцией независимых переменных, т.е.:

|  |  |
| --- | --- |
|  | (2) |

Данную функцию можно использовать для задачи оценки вероятности принадлежности входного вектора к определенному классу. Например, если рассматривается исход по займу, задается переменная *y* со значениями 1 и 0, где 1 означает, что соответствующий заемщик расплатился по кредиту, а 0, что имел место дефолт.

Однако, множественная регрессия игнорирует ограничения на диапазон значений для *y*, так как она не «знает», что переменная отклика бинарна по своей природе. Это неизбежно приведет к модели с предсказываемыми значениями большими 1 и меньшими 0.

Для решения проблемы нужно предсказать непрерывную переменную со значениями на отрезке [0,1] при любых значениях независимых переменных [2]. Это достигается применением следующего регрессионного уравнения (логит-преобразование) в выражении 3:

|  |  |
| --- | --- |
|  | (3) |

где — вероятность того, что произойдет интересующее событие — основание натуральных логарифмов 2,71…; — стандартное уравнение регрессии.

Зависимость, связывающая вероятность события и величину , показана на графике – рисунок 1.1.



Рисунок 1.1 – Логистическая кривая

Существует несколько способов нахождения коэффициентов логистической регрессии [9]. Например, можно применять любые градиентные методы: метод сопряженных градиентов, методы переменной метрики и другие.

**ROC-кривая** (Receiver Operator Characteristic) — кривая, которая наиболее часто используется для представления результатов бинарной классификации в машинном обучении. Название пришло из систем обработки сигналов. Поскольку классов два, один из них называется классом с положительными исходами, второй — с отрицательными исходами. ROC-кривая показывает зависимость количества верно классифицированных примеров одного класса от количества неверно классифицированных примеров другого класса.

В терминологии ROC-анализа первые называются истинно положительным, вторые — ложно отрицательным множеством. При этом предполагается, что у классификатора имеется некоторый параметр, варьируя который, будет получаться то или иное разбиение на два класса. Этот параметр часто называют порогом, или точкой отсечения. В зависимости от него будут получаться различные величины ошибок I и II рода.

В логистической регрессии порог отсечения изменяется от 0 до 1 — это и есть расчетное значение уравнения регрессии. Для понимания сути ошибок I и II рода необходимо рассмотреть таблицу сопряженности 1.1 **(**confusion matrix), которая строится на основе результатов классификации моделью и фактической (объективной) принадлежностью примеров к классам.

Таблица 1.1 – Таблица сопряженности

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | True Positive (TP) | False Positive (FP) |
|  | False Negative (FN) | True Negative (TN) |

Здесь – это ответ алгоритма на объекте, а – истинная метка класса на этом объекте. *TP* — верно классифицированные примеры одного класса (так называемые истинно положительные случаи), *TN* — верно классифицированные примеры второго класса (истинно отрицательные случаи), *FN* — положительные примеры, классифицированные как отрицательные (ошибка I рода), *FP*— отрицательные примеры, классифицированные как положительные (ошибка II рода) [14].

При анализе чаще оперируют не абсолютными показателями, а относительными — долями (rates), выраженными в процентах: доля истинно положительных примеров (True Positives Rate) и доля ложно положительных примеров (False Positives Rate). Они представлены в выражениях 9 и 10:

|  |  |
| --- | --- |
|  | (9) |
|  | (10) |

Логистическая регрессия также имеет [чувствительность](https://wiki.loginom.ru/articles/sensitivity.html) и [специфичность](https://wiki.loginom.ru/articles/specificity.html) модели. Ими определяется объективная ценность любого бинарного классификатора.

Чувствительность (Sensitivity) — это и есть доля истинно положительных случаев (выражение 11):

|  |  |
| --- | --- |
|  | (11) |

Специфичность (Specificity) — доля истинно отрицательных случаев, которые были правильно идентифицированы моделью (выражение 12):

|  |  |
| --- | --- |
|  | (12) |

Модель с высокой чувствительностью часто дает истинный результат при наличии положительного исхода (обнаруживает положительные примеры). Наоборот, модель с высокой специфичностью чаще дает истинный результат при наличии отрицательного исхода (обнаруживает отрицательные примеры).



Рисунок 1.2 – ROC-кривая

ROC-кривая получается следующим образом (рисунок 1.2):

1. Для каждого значения порога отсечения, которое меняется от 0 до 1 с шагом *dx*​ (например, 0,01) рассчитываются значения чувствительности *Se* и специфичности *Sp.* В качестве альтернативы порогом может являться каждое последующее значение примера в выборке.
2. Строится график зависимости: по оси *Y* откладывается чувствительность *Se*, по оси X *-*  — доля ложно положительных случаев.

## Метод дерева решений

Дерево решений — это метод представления решающих правил в иерархической структуре, состоящей из элементов двух типов — узлов и листьев. В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому-либо атрибуту обучающего множества [12].

В простейшем случае, в результате проверки, множество примеров, попавших в узел, разбивается на два подмножества, в одно из которых попадают примеры, удовлетворяющие правилу, а в другое — не удовлетворяющие. Затем к каждому подмножеству вновь применяется правило, и процедура рекурсивно повторяется пока не будет достигнуто некоторое условие остановки алгоритма.

Пример дерева для принятия решения по страхованию клиента приведен на рисунке 1.4.



Рисунок 1.4 – Дерево решений процесса выдачи кредита

Основная сфера применения деревьев решений — поддержка процессов принятия управленческих решений, используемая в статистике, анализе данных и [машинном обучении](https://wiki.loginom.ru/articles/machine-learning.html) [5]. Задачами, решаемыми с помощью данного аппарата, являются:

1. *Классификация* — отнесение объектов к одному из заранее известных классов. Целевая переменная должна иметь дискретные значения.
2. *Регрессия* (численное предсказание) — предсказание числового значения независимой переменной для заданного входного вектора.
3. *Описание объектов* — набор правил в дереве решений позволяет компактно описывать объекты. Поэтому вместо сложных структур, описывающих объекты, можно хранить деревья решений.

Процесс построения деревьев решений заключается в последовательном, рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Процесс разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями.

В настоящее время разработано значительное число алгоритмов обучения деревья решений. Но наибольшее распространение и популярность получили следующие:

* *ID3* (Iterative Dichotomizer 3) — алгоритм позволяет работать только с дискретной целевой переменной, поэтому деревья решений, построенные с помощью данного алгоритма, являются классифицирующими.
* *C4.5* — усовершенствованная версия алгоритма ID3, в которую добавлена возможность работы с пропущенными значениями атрибутов.
* *CART* (Classification and Regression Tree) — алгоритм обучения деревьев решений, позволяющий использовать как дискретную, так и непрерывную целевую переменную, то есть решать, как задачи классификации, так и регрессии.

В ходе построения дерева решений нужно решить несколько основных проблем, с каждой из которых связан соответствующий шаг процесса обучения:

1. Выбор атрибута, по которому будет производиться разбиение в данном узле (атрибута разбиения).
2. Выбор критерия остановки обучения.
3. Выбор метода отсечения ветвей (упрощения).
4. Оценка точности построенного дерева.

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут (признак), по которому это будет сделано. Выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше [6]. Наиболее популярными процедура для данного процесса являются:

1. **Теоретико-информационный критерий**

Как следует из названия, критерий основан на понятиях теории информации, а именно — [*информационной энтропии*](https://wiki.loginom.ru/articles/inform-entropy.html) (выражение 13).

|  |  |
| --- | --- |
|  | (13) |

где *n* — число классов в исходном подмножестве, *Ni*​ — число примеров *i*-го класса, *N* — общее число примеров в подмножестве.

Таким образом, энтропия может рассматриваться как мера неоднородности подмножества по представленным в нём классам. Когда классы представлены в равных долях и неопределённость классификации наибольшая, энтропия также максимальна.

На практике, однако, говорят не об энтропии, а о величине, обратной ей, которая называется информацией. Тогда лучшим атрибутом разбиения будет тот, который обеспечит максимальный прирост информации результирующего узла относительно исходного (выражение 14):

|  |  |
| --- | --- |
|  | (14) |

где *Info(S)*— информация, связанная с подмножеством *S* до разбиения, *Info(SA​)* — информация, связанная с подмножеством, полученными при разбиении по атрибуту *A*.

Таким образом, задача выбора атрибута разбиения в узле заключается в максимизации величины , называемой приростом информации (от англ. gain — прирост, увеличение). Поэтому сам теоретико-информационный подход известен как [*критерий прироста информации*](https://wiki.loginom.ru/articles/info-gain.html). Он впервые был применён в алгоритме ID3, а затем в C4.5 и других алгоритмах.

1. **Статистический подход**

В основе статистического подхода лежит использование индекса Джини. Статистический смысл данного показателя в том, что он показывает — насколько часто случайно выбранный пример обучающего множества будет распознан неправильно, при условии, что целевые значения в этом множестве были взяты из определённого статистического распределения.

Таким образом индекс Джини фактически показывает расстояние между двумя распределениями — распределением целевых значений, и распределением предсказаний модели. Чем меньше данное расстояние, тем лучше работает модель. Индекс Джини может быть рассчитан по формуле 15:

|  |  |
| --- | --- |
|  | (14) |

где *Q* — результирующее множество, *n* — число классов в нём, *pi*​ — вероятность i-го класса (выраженная как относительная частота примеров соответствующего класса).

## Метод случайного леса

RF (random forest с англ. случайный лес) — это множество решающих деревьев. В задаче регрессии их ответы усредняются, в задаче классификации принимается решение голосованием по большинству [3]. Все деревья строятся независимо по следующей схеме:

1. Генерируется случайная подвыборку с повторением размером *N* из обучающей выборки. (Таким образом, некоторые примеры попадут в неё несколько раз, а в среднем , т.е. примерно примеров не войдут в неё вообще).
2. Строится дерево, классифицирующее примеры данной подвыборки, причём в ходе создания очередного узла дерева будет выбираться признак, на основе которого производится разбиение, не из всех *M* признаков, а лишь из *m* случайно выбранных. Дерево строится до полного исчерпания подвыборки и не подвергается процедуре отсечения.
3. Классификация объектов проводится путём голосования: каждое дерево комитета относит классифицируемый объект к одному из классов, и побеждает класс, за который проголосовало наибольшее число деревьев.
4. Оптимальное число деревьев (*n\_estimators*) подбирается таким образом, чтобы минимизировать ошибку классификатора на валидационной выборке [4].

Такая схема построения соответствует главному принципу ансамблирования (построению алгоритма машинного обучения на базе нескольких, в данном случае решающих деревьев): базовые алгоритмы должны быть хорошими и разнообразными (поэтому каждое дерево строится на своей обучающей выборке и при выборе расщеплений есть элемент случайности).

В библиотеке scikit-learn есть такая реализация с помощью класса:

class sklearn.ensemble.RandomForestClassifier(n\_estimators=10, criterion='gini', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0,max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_split=1e-07, bootstrap=True, oob\_score=False, n\_jobs=1, random\_state=None, verbose=0, warm\_start=False, class\_weight=None)

Основные **параметры** данного класса:

1. *n\_estimators* – число деревьев

Чем больше деревьев, тем лучше качество, но время настройки и работы RF также пропорционально увеличиваются [7]. Часто при увеличении n\_estimators качество на обучающей выборке повышается.

1. *max\_features* – число признаков для выбора расщепления.

График качества на тесте от значения этого параметра унимодальный, на обучении он строго возрастает. При увеличении max\_features увеличивается время построения леса, а деревья становятся «более однообразными». По умолчанию он равен sqrt(n) в задачах классификации и n/3 в задачах регрессии [7].

1. *min\_samples\_split* – минимальное число объектов, при котором выполняется расщепление.
2. *min\_samples\_leaf*  – ограничение на число объектов в листьях .

Параметры min\_samples\_leaf и min\_samples\_split, как правило, не очень важные, часто по умолчанию оставляют параметр (2).

1. *max\_depth –* м аксимальная глубина деревьев.

Чем меньше глубина, тем быстрее строится и работает RF. При увеличении глубины резко возрастает качество на обучении, но и на контроле оно, как правило, увеличивается.

1. criterion – критерий расщепления.

Функция для измерения качества разделения. Поддерживаемые критерии: «gini» для примеси Джини и «entropy» для получения информации.

# Решение задачи предсказания взятия клиентом кредита

1.

## Постановка задачи

Для реализации моделей машинного обучения на сайте UCI Machine Learning Repository (репозиторий машинного обучения) [11] был взят набор данных Bank Marketing Data Set (Набор Маркетинговых Данных Банка) [10].

Эти данные связаны с прямыми маркетинговыми кампаниями португальского банковского учреждения. Маркетинговые кампании основывались на телефонных звонках. Часто требовалось более одного контакта с одним и тем же клиентом, для того, чтобы получить доступ, если продукт (банковский срочный депозит) будет (или не будет) подписан.

Существует два набора данных:

1) bank-full.csv со всеми примерами, упорядоченными по дате (с мая 2008 года по ноябрь 2010 года).

2) bank.csv с 10% примеров (4521), случайно выбранных из банка-полный.csv.

Самый маленький набор данных предоставляется для тестирования более сложных в вычислительном отношении алгоритмов машинного обучения (например, SVM).

Цель классификации-предсказать, подпишет ли клиент срочный депозит (переменная y).

Данные набор данных имеет следующую атрибутивную информацию:

1. Входные переменные:

Данные клиента банка:

* age (возраст) – numeric (числовой);
* job (тип работы) – categorical (категориальный): "admin.", "unknown", "unemployed", "management", "housemaid", "entrepreneur", "student", "blue-collar", "self-employed", "retired", "technician", "services";
* marital (семейной положение) - categorical (категориальный): "married", "divorced", "single";
* education (образование) – categorical (категориальный): "unknown", "secondary", "primary", "tertiary";
* default (есть ли кредит в дефолте?) - binary (двоичный код): "yes", "no";
* balance (среднегодовой баланс, в евро) - numeric (числовой);
* housing (есть ли жилищный кредит?) - binary (двоичный код): "yes", "no";
* loan (есть ли личный кредит?) - binary (двоичный код): "yes", "no";

Данные, связанные с последним контактом текущей кампании:

* contact (тип контактной связи) - categorical (категориальный): "unknown","telephone","cellular");
* day (последний контактный день месяца) - numeric (числовой);
* month (последний контактный месяц года) - categorical (категориальный): "jan", "feb", "mar", ..., "nov", "dec";
* duration (длительность последнего контакта, в секундах) - numeric (числовой);

Прочие атрибуты:

* campaign (количество контактов, выполненных в ходе данной кампании и для данного клиента) - numeric (числовой);
* pdays (количество дней, прошедших после последнего контакта с клиентом из предыдущей кампании) - numeric (числовой);
* previous (количество контактов, выполненных до начала данной кампании и для данного клиента) - numeric (числовой);
* poutcome (итоги предыдущей маркетинговой кампании) - categorical (категориальный): "unknown", "other", "failure", "success".
1. Выходная переменная (желаемая цель):
* y (подписался ли клиент на срочный депозит?) - binary (двоичный код): "yes", "no".

## Проведения предобработки данных поставленной задачи

Для начала необходимо загрузить данные из файла bank-full.cvs. Для это используется следующая команда:

X = pd.read\_csv("bank-full.csv")

Для загрузки используется библиотека, которая была загружена в начале:

import pandas as pd

Для просмотра первых 5 записей загруженных данных (рисунок 2.1), необходимо использовать команду:

X.head()



Рисунок 2.1 – Загруженные данные

Далее необходимо проверить наличие пустых значений в наборе данных для каждого признака с помощью библиотеки seaborn:

import seaborn as sns

X.isnull().sum()

sns.heatmap(X.isnull(), cbar = False).set\_title("Пропущенные значения")

Как видно на рисунке 2.2 набор данных не имеет пропущенных значений.



Рисунок 2.2 – Пропущенные значения набора данных

Для того, чтобы проверить число строк и столбцов в наборе данных, используется команда:

X.shape

В результате было получено 45211 строк и 17 столбцов.

Далее необходимо посмотреть, как распределены значения в зависимом столбце (рисунок 2.3), т.е. подписался ли клиент на срочный депозит или нет:

X['y'].value\_counts(dropna=False)



Рисунок 2.3 – Распределение значений в зависимом столбце

Далее необходимо посмотреть с какой точностью будет работать модель, если будет предсказывать всем клиентам, что они не подпишут срочный депозит. Для этого нужно разделить количество отрицательных значений в зависимом признаке на количество всех записей в наборе данных (d=39922/45211). В результате было получено значение 0.8830151954170445. Так как выборка несбалансированная, построенная модель будет считаться хорошей, только если ее точность превзойдет это значение.

С помощью команды X.info() необходимо проверить типы каждого признака (рисунок 2.4). Как видно такие признаки, как: age (возраст), balance (средний годовой баланс, в евро), duration (длительность последнего контакта, в секундах), campaign (количество контактов, выполненных в ходе данной кампании и для данного клиента), pdays (количество дней, прошедших после последнего контакта с клиентом из предыдущей кампании), previous (количество контактов, выполненных до начала данной кампании и для данного клиента), - содержат числовые значение, которые хорошо сочетаются с алгоритмами машинного обучения, в отличие других признаков с текстовыми данными, поэтому их нужно преобразовать.



Рисунок 2.4 - Информация о наборе данных

Поля дефолт (есть ли кредит в дефолте?), жилье (есть ли жилищный кредит?), кредит (есть ли личный кредит?) можно закодировать обычным способом (0 и 1):

X['y'] = pd.factorize(X['y'])[0]

X['default'] = pd.factorize(X['default'])[0]

X['housing'] = pd.factorize(X['housing'])[0]

X['loan'] = pd.factorize(X['loan'])[0]

Объектные (текстовые) переменные: профессия клиента, семейное положение, образование, тип контактной связи, итоги предыдущей маркетинговой кампании, - необходимо перевести в числовой формат с помощью one-hot кодирования:

X = pd.concat([X, pd.get\_dummies(X['job'], prefix="job")], axis=1)

X = pd.concat([X, pd.get\_dummies(X['marital'], prefix="marital")], axis=1)

X = pd.concat([X, pd.get\_dummies(X['education'], prefix="education")], axis=1)

X = pd.concat([X, pd.get\_dummies(X['contact'], prefix="contact")], axis=1)

X = pd.concat([X, pd.get\_dummies(X['poutcome'], prefix="poutcome")], axis=1)

В результате каждый признак перекодируется в несколько новых бинарных признаков. Поэтому старые нечисловые столбцы нужно удалить:

X.drop(['job', 'marital', 'education', 'contact','poutcome'], axis=1, inplace=True)

После выполнены действий нужно убедиться в отсутствии нечисловых признаков (рисунок 2.5).



Рисунок 2.5 – Состав набора данных после перекодировки

## Обучение модели логистической регрессии

После того, как набор данных был преобразован, необходимо приступить к обучению. Для начала нужно задать входы и выходы модели:

y = X['y']

X = X.drop(('y'), axis=1)

Далее нужно сформировать из набора данных тестовую и обучающие выборки с помощью библиотеки sklearn.model\_selection:

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=20)

Для обучения логистической регрессии необходимо использовать библиотеку sklearn.linear\_model:

from sklearn.linear\_model import LogisticRegression

clf = LogisticRegression()

clf.fit(X\_train, y\_train);

Далее необходимо посчитать точность (долю правильных ответов) на тестовой выборке:

scoreLogicRegres = clf.score(X\_test, y\_test)

Точность логистической регрессии равна *0.8997014265177485,* что выше, чем baseline 0.8830151954170445.

А также вычислить метрику точности ROC-AUC с помощью библиотеки sklearn.metrics:

y\_pred = clf.predict\_proba(X\_test)[:, 1]

from sklearn.metrics import roc\_auc\_score

roc\_auc\_scoreLogicRegres = roc\_auc\_score(y\_test, y\_pred)

Метрика точности ROC-AUC логистической регрессии равно *0.8799909744748209.*

## Обучение модели случайного леса

Для построения модели случайного леса также сначала задаются входы и выходы и формируются тестовая и обучающая выборки.

Чтобы обучить модель случайного леса используется библиотека sklearn.ensemble:

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators = 10, max\_depth=5, random\_state=21)

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

Затем высчитывается точность модели и метрика точности ROC-AUC:

scoreRandomForest = rfc.score (X\_test, y\_test)

from sklearn.metrics import roc\_auc\_score

roc\_auc\_scoreRandomForest = roc\_auc\_score(y\_test, y\_pred)

Точность случаного леса равна *0.8984850160345018,* что ниже, чем у логистической регрессии.

Метрика точности ROC-AUC случайного леса равно *0.8756026174500813.*

## Обучение модели дерева решений

Для построения модели дерева решений также сначала задаются входы и выходы и формируются тестовая и обучающая выборки.

Чтобы обучить модель случайного леса используется библиотека sklearn.tree:

from sklearn.tree import DecisionTreeClassifier, plot\_tree

clf = DecisionTreeClassifier(max\_depth=3)

clf = clf.fit(X\_train, y\_train)

После обучения модели можно построить дерево решений (рисунок 2.6) с помощью библиотека pyplot:

 plt.figure(figsize=((20,13)))

plot\_tree(clf,

 filled=True,

 feature\_names=list(X\_train.columns),

 class\_names=list(X\_train.columns),

 rounded=True)

plt.show()



Рисунок 2.6 – Дерево решений

Затем высчитывается точность модели и метрика точности ROC-AUC:

scoreTree = clf.score (X\_test, y\_test)

from sklearn.metrics import roc\_auc\_score

*roc\_auc\_scoreTree = roc\_auc\_score(y\_test, y\_pred)*

Точность дерева решений равна *0.9052305650779608,* что является самым высоким значением из трех рассмотренных моделей.

Метрика точности ROC-AUC дерева решений равно *0.8756026174500813*.

На рисунке 2.7 представлены результаты обучения логистической регрессии, случайного леса и дерева решений. Как видно из полученных данных, для данной выборки лучше работает дерево решений.



Рисунок 2.7 – Результаты обучения

1.

# ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта была выполнена главная цель: реализованы такие алгоритмы и методы машинного обучения, как логистическая регрессия, случайный лес и дерево решений.

А также были выполнены следующие задачи:

1. Применен метод логистической регрессии для поставленной задачи;
2. Применен метод случайного леса для поставленной задачи;
3. Применен метод дерева решений для поставленной задачи;
4. Осуществлено сравнение точности данных методов при решении задачи банковского маркетинга.

Таким образом, после реализации разных моделей машинного обучения был сделан вывод, что в задаче банковского маркетинга лучше себя показал метод построения дерева решений.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Васильев Н. П., Егоров А. А. Опыт расчета параметров логистической регрессии методом Ньютона–Рафсона для оценки зимостойкости растений //Математическая биология и биоинформатика. – 2011. – Т. 6. – №. 2. – С. 190-199.
2. Григорьев С. Г., Лобзин Ю. В., Скрипченко Н. В. Роль и место логистической регрессии и ROC-анализа в решении медицинских диагностических задач //Журнал инфектологии. – 2016. – Т. 8. – №. 4. – С. 36-45.
3. Ивкина М. С. Решение задачи классификации на основе случайного леса //Редколлегия сборника. – 2018. – С. 79.
4. Картиев С. Б., Курейчик В. М. Алгоритм классификации, основанный на принципах случайного леса, для решения задачи прогнозирования //Программные продукты и системы. – 2016. – №. 2 (114).
5. Михеев М. Ю. и др. Применение «дерева решений» для анализа состояния сложных систем //Труды Международного симпозиума «Надежность и качество». – 2012. – Т. 2.
6. Некрасов М. В. Применение метода" дерево решений" при принятии инвестиционных решений //Экономика и управление в XXI веке: тенденции развития. – 2013. – №. 10. – С. 171-175.
7. Рахматова А. Ю. и др. Точность методов Случайный лес и Многослойный персептрон в задаче прогнозирования исходов детских ишемических инсультов //Уральский медицинский журнал. – 2017. – №. 10. – С. 58-62.
8. Снегова Е. Г. Применение метода логистической регрессии для прогнозирования вероятности дефолта при экспресс-кредитовании //Национальные интересы: приоритеты и безопасность. – 2013. – №. 5.
9. Сорокин А. С. Построение скоринговых карт с использованием модели логистической регрессии //Интернет-журнал науковедение. – 2014. – №. 2 (21).
10. Bank Marketing Data Set [Электронный ресурс] // Репизиторий набора данных: свободный репозиторий. URL: [https://archive.ics.uci.edu/ml/datasets/Bank+Marketing](https://archive.ics.uci.edu/ml/datasets/Bank%2BMarketing). Дата обращения: 19.06.2020.
11. UC Irvine Machine Learning Repository [Электронный ресурс] // Репизиторий набора данных: свободный репозиторий. URL: <https://archive.ics.uci.edu/ml/index.php>. Дата обращения: 19.06.2020.
12. Деревья решений: общие принципы [Электронный ресурс] // Аналитическая платформа Loginom: свободная. URL: https://loginom.ru/blog/decision-tree-p1. Дата обращения: 19.06.2020.
13. Как легко понять логистическую регрессию [Электронный ресурс] // Habr: свободный. URL: <https://habr.com/ru/company/io/blog/265007/>. Дата обращения: 19.06.2020.
14. Логистическая регрессия и ROC-анализ — математический аппарат [Электронный ресурс] // Аналитическая платформа Loginom: свободная. URL: https://loginom.ru/blog/logistic-regression-roc-auc. Дата обращения: 19.06.2020.