DOI 10.37882/2223-2966.2025.02.40

СРАВНЕНИЕ МЕТРИК БИНАРНОЙ КЛАССИФИКАЦИИ ДЛЯ ОЦЕНКИ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

COMPARISON OF BINARY CLASSIFICATION METRICS FOR EVALUATING THE EFFECTIVENESS OF MACHINE LEARNING ALGORITHMS

E. Shirokova

Summary. The article compares various binary classification metrics to evaluate the effectiveness of machine learning algorithms. Various approaches are considered, such as the support vector machine method, the decision tree method, the random forest method, and the gradient boosting method. The advantages and disadvantages of metrics in the case of balanced and unbalanced data are analyzed, as well as examples of their implementation using the Python Scikit-learn library.

Keywords: machine learning, data classification, class imbalance.

Широкова Екатерина Васильевна
Кандидат физико-математических наук, доцент, Калужский филиал Московского Государственного Технического Университета имени Н.Э. Баумана
kate-info@inbox.ru

Аннотация. В статье проводится сравнение различных метрик бинарной классификации для оценки эффективности алгоритмов машинного обучения. Рассматриваются различные подходы, такие как метод опорных векторов, метод деревьев решений, метод случайного леса и метод градиентного бустинга. Анализируются достоинства и недостатки метрик в случае сбалансированных и несбалансированных данных, а также приводятся примеры их реализации с помощью библиотеки Scikit-learn языка Python.

Ключевые слова: машинное обучение, классификация данных, дисбаланс классов.

Введение

Задача бинарной классификации, является одной из основных задач машинного обучения. В банковской сфере она применяется во всех ключевых областях: прогнозирование вероятности дефолта заёмщика; определение реакции клиента на предложение о продаже дополнительных услуг; предсказание возможности возврата просроченного кредита; детектирование мошеннических операций. В маркетинге эта задача заключается в определении потенциальных покупателей товаров и услуг. В медицине — это один ин инструментов постановки диагноза и т.д.

Бинарная классификация — это процесс отнесения объекта к одному из двух заранее определённых классов. Имеется множество объектов $N = \{n_1, n_2, \ldots, n_n\}$, каждый из которых имеет m признаков $(x_1, x_2 \ldots, x_m)$. Признаки могут быть числовыми или нечисловыми (категориальными). Известно классовое распределение некоторых объектов из исходного множества. Остальные объекты имеют неизвестную классовую принадлежность. Возникает необходимость создать алгоритм, способный определить для произвольного объекта из исходной выборки класс, к которому этот объект должен быть отнесён.

Эффективность бинарной классификации часто измеряется с помощью показателей, предназначенных для устранения недостатков в точности классификации. Например, хорошо известно, что точность классификации является неподходящим показателем для задач классификации редких событий, таких как выявление спама, обнаружение мошенничества, прогнозирование количества кликов, поиск текста в приложениях [1, 2, 3] и др. Вместо этого используются альтернативные показатели, лучше адаптированные к несбалансированной классификации. Важный теоретический вопрос, касающийся показателей, используемых в бинарной классификации, заключается в том, насколько они эффективны и определение оптимальных функций для принятия решений.

Алгоритмы бинарной классификации данных

В качестве классификатор будут использованы метод логистической регрессии, метод опорных векторов, метод случайного леса и метод градиентного бустинга. Все они реализованы с помощью библиотеки машинного обучения Scikit-Learn, написанной на языке Python.

Логистическая регрессия — это один из наиболее простых и эффективных методов бинарной классификации, который использует логистическую функцию для предсказания вероятности принадлежности к одному из двух классов. Ее общее назначение состоит в анализе связи между несколькими $(x_1, x_2, \ldots, x_m)$ независимыми переменными (называемыми также регрессорами или

предикторами) и зависимой переменной *y*. Предполагается, что зависимая переменная является линейной функцией независимых переменных, т.е.:

$$y=a+b_1x_1+b_2x_2+\ldots+b_mx_m,$$

где b_1, b_2, …, b_m — это коэффициенты регрессии. Далее функция логистического преобразования (сигмоидальная функция) преобразует эти коэффициенты в вероятность принадлежности наблюдения к одному из двух классов.

В логистической регрессии мы используем концепцию порогового значения, которое определяет вероятность либо 0, либо 1. Например, значения выше порогового значения стремятся к 1, а значения ниже пороговых значений стремятся к 0. Для оценки параметров модели бинарной логистической регрессии используется метод максимального правдоподобия. Этот метод минимизирует разницу между наблюдаемыми и прогнозируемыми значениями вероятности для каждого наблюдения.

Преимущества логистической регрессии в машинном обучении включают в себя простоту и интерпретируемость модели, высокую скорость обучения и предсказания, а также возможность использования для бинарной классификации. Кроме того, логистическая регрессия хорошо работает с линейно разделимыми данными и хорошо масштабируется на больших объемах данных. Однако у этого метода также есть недостатки, включая то, что он неспособен моделировать сложные нелинейные зависимости, он чувствителен к выбору признаков и предобработке данных, а также имеет ограниченную способность к решению проблемы мульти классовой классификации. В целом, логистическая регрессия является мощным инструментом для базового анализа данных, но может не подойти для сложных задач классификации.

Метод опорных векторов (SVM): SVM является мощным алгоритмом для бинарной классификации, который находит оптимальную разделяющую гиперплоскость между двумя классами. Основная идея метода опорных векторов заключается в поиске наиболее информативных признаков, которые определяют принадлежность объекта к тому или иному классу. Алгоритм находит опорные векторы — объекты с наибольшим вкладом в разделение классов. Существует несколько модификаций метода опорных векторов, таких как линейный SVM, полиномиальный SVM, радиальная базисная функция (RBF) SVM и другие. Выбор конкретной модификации зависит от характера данных и требуемой точности модели.

Суть метода «Случайный лес» (Random Forest) заключается в том, что он создаёт множество решающих деревьев и использует их для предсказания классов объектов. Каждое дерево строится на случайном подмножестве обучающих данных и случайном подмножестве признаков. В результате каждое дерево в ансамбле получается немного разным, что позволяет уменьшить эффект переобучения и повысить качество предсказаний. После создания всех деревьев в ансамбле для каждого объекта данных проводится голосование по всем деревьям, и наиболее популярный класс становится предсказанным классом [4]. Преимущество метода случайного леса состоит в том, что он позволяет обрабатывать большие объёмы данных с высокой размерностью и обладает высокой точностью. Однако этот метод сложнее интерпретировать по сравнению с логистической регрессией.

Градиентный бустинг — это метод машинного обучения, который создаёт модель прогнозирования в виде ансамбля упрощённых прогнозирующих моделей, обычно деревьев решений. Основная цель алгоритмов обучения с учителем — определить функцию потерь и минимизировать её. В случае градиентного бустинга функция потерь может быть среднеквадратичной ошибкой (MSE). Используя градиентный спуск и изменяя прогнозы на основе скорости обучения (learning rate), происходит минимизация MSE. Таким образом, алгоритм постоянно улучшает прогнозы, стремясь к нулевым отклонениям и получению наиболее точных предсказаний. К основным достоинствам этого метода можно отнести способность эффективно находить нелинейные зависимости в данных различной природы. Наибольшую популярность приобрели три модификации этого метода: CatBoost, LightGBM и XGBoost. Несмотря на то, что ранее они отличались довольно сильно, к настоящему моменту все три модификации успели скопировать друг у друга много хороших идей. Поэтому выбор конкретного алгоритма для конкретного набора данных носит эмпирический характер.

Метрики бинарного анализа

Любая модель бинарной классификации может ошибаться, поэтому все объекты выборки разбиваются на четыре типа, образуя матрицу ошибок (confusion matrix). Матрица ошибок (также известная как матрица неточностей) — это способ визуализации для оценки качества классификаторов в машинном обучении с учителем. Она используется для бинарной и много классовой классификации.

Все объекты выборки содержат информацию о количестве верно классифицированных объектов (True Positives, TP) и неверно классифицированных объектов (False Positives, FP), а также о количестве верно неклассифицированных объектов (True Negatives, TN) и неверно неклассифицированных объектов (False Negatives, FN) в виде таблицы 1:

Таблица 1.

Матрица ошибок

	y=1	y=0
predict (x)=1	TP (True Positive)	FP (False Positive)
predict (x)=0	FN (False Negative)	TN (True Negative)

В зависимости от соотношения *TP, FP, TN* и *FN*, можно сделать вывод о том, насколько хорошо работает модель.

По результатам матрицы ошибок можно вычислять различные меры качества модели классификации, такие как долю правильных ответов (*ACC*), точность (*Precision*) и полноту (*Recall*) соответственно:

$$ACC = \frac{TP + TN}{TP + FP + FN + TN},$$

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN}$$

Точность определяет долю объектов, действительно принадлежащих данному классу, относительно всех объектов, которые система отнесла к этому классу. В то время как полнота показывает, какую долю объектов, реально относящихся к положительному классу, мы предсказали верно. Надо отметить, что в случаях несбалансированных выборок, *ACC* является неинформативным показателем, в отличие от *Precision* и *Recall*. Например, мы хотим создать модель для определения мошеннических кредитных заявок, которые составляют только 1 % от всех заявок. Если использовать простой классификатор, который определяет все заявки как «не мошеннические», его точность составит 99 %, а процент ошибок — 1 %. Хотя эти показатели выглядят хорошими, такая модель будет бесполезна для бизнеса, так как она не обнаруживает мошеннические заявки.

Scikit-Learn предоставляет несколько функций для вычисления показателей классификатора, включая точность и полноту. Но можно и отдельно рассчитать компоненты матрицы ошибок с помощью функции *sklearn. metrics. confusion_matrix () [5].*

Во время настройки параметров алгоритма часто оптимизируют общий показатель, улучшение которого ожидается на тестовом наборе данных. Это мера, которая объединяет Precision и Recall в общий критерий качества и представляет собой их среднее гармоническое значение.

$$F_{beta} = \left(1 + \beta^2\right)\frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

Значение β — вес точности. При полноте и точности равных единице, F_{beta}-мера достигает максимума, и близка к нулю, если один из аргументов близок к нулю.

В библиотеке scikit-learn есть удобная функция *sklearn. metrics. Classificationreport ()* [5], которая возвращает показатели *recall, precision* и F_{beta}*-мера* для каждого класса, а также количество экземпляров каждого класса.

ROC-кривая (receiver operating characteristic) — еще один распространенный инструмент, используемый в бинарных классификаторах. График, используемый для оценки качества бинарной классификации, показывает соотношение между долей правильно классифицированных объектов *TPR* (True Positive Rate) и долей ошибочно классифицированных объектов *FPR* (False Positive Rate).

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Анализ классификаций с использованием *ROC*-кривых известен как *ROC*-анализ. Для более достоверной интерпретации результатов *ROC*-анализа необходимо определить порог классификации, при котором результат либо 1, либо 0. В общем случае порогом является значение 0.5. А оценка площади под ROC-кривой определяет так называемый *AUC*-анализ (Area Under Curve). В идеальном случае мы получаем значение площади равное 1. Критерий *AUC* может рассматриваться как вероятность того, что случайно выбранный положительный объект будет оценён моделью выше, чем случайно выбранный отрицательный объект.

Данный вид анализа получил широкую популярность в машинном обучении, т.к. отображает общую производительность модели в целом. Однако может вводить в заблуждение при сильном дисбалансе классов [6]. Поэтому часто можно встретить метрику *AUC-PR*, которая рассчитывает площадь под кривой *Precition-Recall*, аналогично *AUC-ROC*. С помощью этого графика кривой *Precition-Recall* можно принять взвешенное решение в рамках классической проблемы выбора между точностью и полнотой. Понятно, что увеличение полноты ведёт к снижению точности. Определение уровня полноты, при котором точность начинает стремительно снижаться, поможет установить оптимальное пороговое значение и создать более эффективную модель.

В библиотеке scikit-learn критерий *AUC* реализует функция *sklearn.metrics.roc_auc_score (),* а площадь под кривой — *Precition-Recall sklearn. metrics. average_ precision_score ()* [5].

Коэффициент корреляции Мэтьюса *MCC*— это показатель качества бинарных классификаторов, который учитывает истинные и ложные классификации и является сбалансированной мерой, подходящей для использования в условиях дисбаланса классов. *MCC* изменяется в диапазоне от –1 до 1, где 1 указывает на идеальную классификацию без ложных классификаций, 0 соответствует случайному предсказателю, а –1 обозначает полное расхождение между фактом и предсказанием.

Корреляцию между предсказанными классами и реальностью можно рассчитать на основе значений из матрицы ошибок следующим образом:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}$$

Sklearn.metrics. matthews_corrcoef() — программная реализация *MCC* в библиотеке scikit-learn [5].

Еще одна часто используемая метрика производительности — это функция потерь *Log Loss* (также известная как Cross Entropy Loss). Она вычисляется путём сравнения предсказанных моделью вероятностей классов с истинными значениями и минимизируется в процессе обучения модели. Расчетная формула в общем виде выражается:

$$Log\,Loss = -\frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot \ln(p_i),$$

где n — количество примеров, y_i— истинный класс i-го примера, p_i— предсказанная вероятность. Чем меньше значение, тем лучше модель.

В отличие от других метрик, таких как *Accuracy* или F_{beta}, *Log Loss* менее чувствителен к ошибкам в редких классах. Это делает ее более подходящей для анализа данных с большим количеством выбросов. Чем более определена наша модель в том, что наблюдение положительно, когда оно на самом деле положительно, тем ниже ошибка. Но это не линейная зависимость, поэтому могут возникнуть сложности при ее оптимизации.

Ограничения этой модели связаны с тем, что *Log Loss* может быть неустойчивой к выбросам, т.е. если в данных есть большие отклонения, то эта оценка может быть менее точной.

В библиотеке scikit-learn функция потерь рассчитывается с помощью *sklearn.metric.log_loss ()* [5].

Информация о наборе данных

Данные, используемые в данном работе, основаны на маркетинговом исследовании португальского банка [7]. В результате опроса клиентов была собрана информация, в которой учтено внес ли клиент средства на банковский срочный депозит. Набор данных содержит 17 записей о каждом опрошенном, из которых 5 являются числовыми, а остальные — категориальными. Первые 16 составляют ключевую информацию о клиенте (возраст, должность, образование, есть ли кредит на жилье и т.д.). В последнем отражен сам результат в виде одного из двух ответов «да» или «нет». Для сравнения использовалось два набора данных. Первый набор состоит из 4000 строк с информацией о клиентах, которые так и не открыли депозит, и 360 строк данных о клиентах, которые открыли. Этот сет данных можно назвать несбалансированным, т.к. в нем содержится всего 9 % данных меньшего класса. Для сравнения последующих вычислений был использован второй сет из 8000 данных, в котором классы разбиты в равном отношении. Оба датасета подверглись предобработке, в результате чего все категориальные данные были заменены числовыми.

Результат моделирования

В таблицах 2 и 3 приведены расчеты показателей различных метрик классификации как для сбалансированного, так и для несбалансированного наборов данных. Для этого были задействованы следующие алгоритмы: логистическая регрессия, метод опорных векторов, случайный лес и алгоритм градиентного бустинга. Все они были реализованы с помощью Python-библиотеки Scikit-learn [8]. Для каждого из них были подобраны наилучшие параметры. Для функции *LogisticRegression* установлены решатель (*solver= 'liblinear'*) и значение коэффициента силы связи между независимой и зависимой переменными (*C=7*). Функция *LinearSVC ()* реализует линейный метод опорных векторов *Linear Support Vector Machines*. В данной работе было принято решение оставить все параметры этой функции по умолчанию. Для построения модели случайного леса используется функция *RandomForestRegressor ()*. В качестве параметра, определяемого минимальное количество выборок, необходимое для листового узла, выбрано значение 6 (*min_samples_leaf=6*). Количество деревьев равно 20 (*n_estimators=20*). *GradientBoostingClassifier* — это реализация алгоритма градиентного бустинга, предоставленная библиотекой Scikit-Learn. В этом алгоритме установлены параметры: 500 деревьев (*n_estimators=500*), минимальное количество выборок для внутреннего узла при разделении на дополнительные узлы (min_samples_split=3), минимальное количество выборок листа после разделения (min_samples_leaf=5), сохранение предыдущих результатов обучения и использование их для инициализации новых итераций (warm_start= True).

Метод случайного леса выигрывает по большинству основных показателей качества алгоритма машинного обучения, таких как *ROC AUC* = 0,97, *PR AUC | Average* = 0,97, *Log loss* = 0,19. По остальным метрикам уступает не-

Таблица 2.

Рассчитанные значения различных метрик оценки качества классификаторов для набора сбалансированных данных [7]

	Logistic Regression	LinearSV	HistGradient Boosting	RandomForest Regressor
TN	262	264	282	284
FP	39	18	17	26
FN	29	29	19	15
TP	270	289	282	275
Recall	0.903	0.908	0.909	0.948
Precision	0.8732	0.941	0.943	0.913
Auc	0.954	0.922	0.936	0.975
Accuracy	0.886	0.921	0.940	0.931
F1 score	0.888	0.924	0.940	0.930
PR AUC | Average	0.947	0.904	0.915	0.971
Log loss	0.286	2.823	2.162	0.194
MMC	0.773	0.843	0.880	0.863

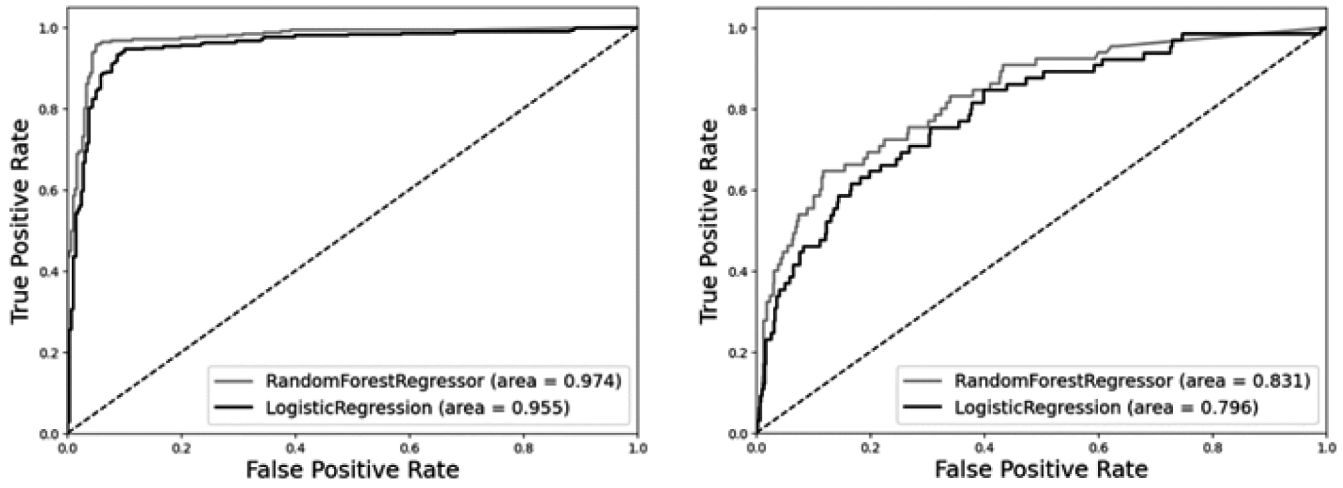
Таблица 3.

Рассчитанные значения различных метрик оценки качества классификаторов для набора несбалансированных данных [7]

	Logistic Regression	LinearSV	HistGradient Boosting	RandomForest Regressor
TN	1192	1203	1184	1179
FP	8	3	16	17
FN	60	58	50	58
TP	5	1	15	11
Recall	0.076	0.017	0.230	0.159
Precision	0.384	0.250	0.483	0.392
Auc	0.795	0.507	0.608	0.864
Accuracy	0.946	0.951	0.947	0.940
F1 score	0.128	0.031	0.312	0.226
PR AUC | Average	0.235	0.050	0.151	0.306
Log loss	0.174	1.738	1.880	0.161
MMC	0.153	0.054	0.310	0.224

значительно. На рис. 1 кривые *ROC AUC* для трех лучших алгоритмов.

Статистика исследований [9] говорит, что 79 % успешных продаж могут быть достигнуты при обращении только к половине клиентов. Поэтому важно предсказать, какие клиенты с большей вероятностью подпишутся, чтобы банк мог нацелиться на этих клиентов и улучшить их коэффициент конверсии. В связи с этим очень важно выявить основные параметры о клиенте из всех доступных для анализа. С точки зрения машинного обучения это вопрос отбора наиболее значимых признаков (feature selection). На рис. 2. представлена гистограмма признаков набора данных, из которой видно, что среднегодовой баланс и длительность последнего звонка с клиентом играет ключевую роль в прогнозировании.



а) б)

Рис. 1. Рассчитанные значения *ROC*-кривых с помощью классификаторов логистической регрессии и случайного леса для: а) сбалансированного и б) несбалансированного наборов данных

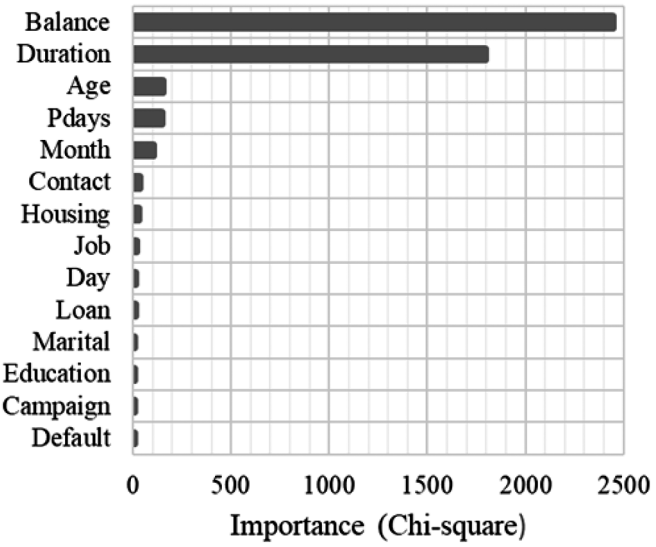


Рис. 2. Гистограмма признаков по критерию Chi-square

Выводы

Проведен сравнительный анализ алгоритмов машинного обучения. Рассмотрены результаты двух наборов данных о потенциальных клиентах банка. Для набора сбалансированных данных все представленные алгоритмы показывают достаточно хорошую эффективность классификаторов. Лучшими являются алгоритм случайного леса и логистической регрессии ROC AUC (от 0,95). В случае несбалансированных данных (9 % данных меньшего класса) необходимо использовать доступные варианты балансировки (взвешивание классов, увеличение выборки, уменьшение выборки, генерация синтетических данных). Метод случайного леса превосходит другие модели в предсказаниях о склонности клиента к размещению депозита, что имеет большое значение для привлечения потенциальных клиентов и формирования депозитной политики банка. Особенно это заметно, в случае несбалансированных данных, где *ROC AUC* = 0,84, *PR AUC* | *Average* =0,30, а качество модели на основе вероятности правильных предсказаний *Log loss* = 0,16.

Результаты моделирования могут быть полезны при выборе правильной стратегии в управлении процессами принятия решений различных сфер деятельности.

ЛИТЕРАТУРА

1. Drummond C., Holte R.C. Severe class imbalance: Why better algorithms aren't the answer? // Machine Learning. 2005. P. 539–546.
2. Gu, Q., Zhu, L., Cai, Z. Evaluation measures of the classification performance of imbalanced data sets // Computational Intelligence and Intelligent Systems. 2009. P. 461–471.
3. Haibo H., Garcia E.A. Learning from imbalanced data// Knowledge and Data Engineering. 2009. N 9. P.1263–1284.
4. Breiman W. Random Forests // Machine Learning. 2001. N 45. P. 5–32.
5. Scikit-learn: Scikit-learn. Машинное обучение в Python. [Сайт]. URL: https://scikit-learn.ru/stable/modules/model_evaluation.html (Дата обращения 20.11.2024).
6. Saito T., Rehmsmeier M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets // PLoS ONE. 2015. N 10. URL: https://www.sci-hub.ru/10.1371/journal.pone.0118432 (Дата обращения 25.11.2024).
7. UC Irvine machine Learning Repository: [Сайт]. URL: https://archive.ics.uci.edu/dataset/222/bank+marketing (Дата обращения 12.11.2024).
8. Scikit-learn: Scikit-learn. Машинное обучение в Python. [Сайт]. URL: https://scikit-learn.ru/stable/supervised_learning.html#supervised-learning (Дата обращения 20.11.2024).
9. Moro S., Cortez P., Rita P.A data-driven approach to predict the success of bank telemarketing // Decision Support Systems. 2014. N 62. P. 22-31.

Журнал «Современная наука: актуальные проблемы теории и практики»