Компьютерные инструменты в образовании, 2018
№ 5: 5–23
УДК: 004.855:654.07
http://ipo.spb.ru/journal
doi:10.32603/2071-2340-2018-5-5-23

ПРЕДСКАЗАНИЕ ОТТОКА АБОНЕНТОВ: СРАВНЕНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Арзамасцев С. А., Бгатов М. В., Картышева Е. Н., Деркунский В. А., Семенчиков Д. Н.

Санкт-Петербургский государственный университет, Санкт-Петербург, Россия

Аннотация

Чтобы оставаться конкурентноспособным сегодня в телекоммуникационном бизнесе, необходимо определять клиентов, которые недовольны предоставляемыми услугами, поэтому прогнозирование оттока стало актуальной проблемой в данной сфере. В этой статье рассмотрены основные современные алгоритмы машинного обучения, которые применялись для решения этой задачи, включая дерево принятия решений (DT — Decision Trees), наивный байесовский классификатор (NB — Naive Bayes Classifier), случайный лес (RF — Random Forest), искусственные нейронные сети (NN — Artificial Neural Network), метод k-ближайших соседей (KNN — K-Nearest Neighbors), линейный дискриминантный анализ (LDA — Linear Discriminant Analysis), метод опорных векторов (SVM — Support Vector Machine) и их ансамблирование (бэггинг и бустинг) с целью продемонстрировать превосходство новой технологии CatBoost в мерах эффективности классификаторов. Для достижения цели была проведена классификация данных и выявлены конкретные преимущества метода CatBoost в сравнении с другими на основе полученных результатов.

Для проведения исследования нами были проанализированы четыре базы данных: 3 датасета находятся в открытом доступе и 1 датасет, предоставленный российской мобильной компанией. Зачастую размерность этих баз данных высока, что приводит к ряду проблем (в том числе несбалансированности классов, корреляции параметров), которые решаются методом уменьшения размерности: метод главных компонент (PCA — Principal Component Analysis).

Полученные результаты сравниваются между собой, а также с результатами, представленными другими исследователями на основе открытых баз данных. Эффективность классификаторов оценивается с помощью таких мер, как площадь под кривой (AUC), точность, F_1-мера и время.

Ключевые слова: *анализ данных, машинное обучение, балансировка данных, обработка данных, ансамбли моделей.*

Цитирование: Арзамасцев С. А., Бгатов М. В., Картышева Е. Н., Деркунский В. А., Семенчиков Д. Н. Предсказание оттока абонентов: сравнение методов машинного обучения // Компьютерные инструменты в образовании, 2018. № 5. С. 5–23. doi:10.32603/2071-2340-2018-5-5-23

1. ВВЕДЕНИЕ

С развитием бизнеса [6] высокое качество услуг перестало быть единственным фактором, гарантирующим предприятию стойкие позиции на рынке, так как зачастую клиент оценивает компанию по многим критериям, от которых зависит его лояльность: связь отделов (продаж, маркетинга и т. д.) и их успешная совместная работа, определение желаний клиентов на всех стадиях взаимодействия, удачная реклама и др. Все это помогает создать клиентоориентированный подход [7] (CRM — Customer Relationship Management), который включает в себя:

- Базу данных о всех участвующих лицах (клиенты, деловые партнеры и др.).
- Пути анализа информации в базе данных.
- Стратегию применения анализа для улучшения бизнеса (сохранение текущих клиентов и привлечение новых).
- Проверку эффективности методов.

Имея детальную информацию о клиенте, компания получает возможность вовремя предпринять необходимые меры по предотвращению оттока и увеличению притока, например, бонусные системы, программы лояльности, акции и т. д. Впоследствии это приводит к увеличению денежного капитала и развитию компании. В связи с этим возникает задача предсказания поведения клиента [2].

Одна из типичных задач, возникающих в процессе определения поведения абонента — это определение принадлежности клиента к одному из двух классов: лояльных к компании и склонных к уходу (задача бинарной классификации). С помощью современных методов машинного обучения и первичной обработки данных эта задача эффективно решается. Поскольку предсказание поведения клиента является одной из наиболее актуальных проблем в CRM, нами была выбрана именно эта область для исследования эффективности нового метода CatBoost в сравнении с другими.

В статье [6] представляется использование науки о данных для прогнозирования оттока клиентов на датасете пакистанской телекоммуникационной компании. В частности, в ней представлены некоторые из методов, использованных в нашей работе — это DT, SVM и NN. Помимо классификации клиентов, получены дополнительные результаты с помощью анализа причинно-следственных связей для выявления наиболее влияющих факторов.

В статье [7] также исследуется проблема оттока клиентов. Анализ производится на базе реальных данных о покупателях в розничной торговле. В работе используются следующие методы классификации: DT, логистическая регрессия, KNN, RF и Bagging DT. Сравниваются результаты работы методов по оценкам AUC и Accuracy.

В статье [2] описывается подход к решению проблемы оттока клиентов телекоммуникационных компаний. С помощью метрик Precision, Recall и F_1-меры проводится сравнительный анализ методов DT, RF, KNN и NB. Проводится балансировка классов, а также используется решетчатый поиск (GS) для выбора параметров с наилучшим результатом по оценкам.

В статье [10] разрабатывается собственная модель оттока клиентов интернет-магазина, имеющего огромную клиентскую базу. На первом этапе проводится анализ корреляции данных, дисперсионный анализ, PCA для уменьшения количества имеющихся параметров. Затем используются методы классификации с применением кросс-валидации: NB, SVM, DT, RF и логистическая регрессия. Используются метрики Accuracy и AUC для выбора лучших способов классификации.

2. СПОСОБЫ ОЦЕНКИ ЭФФЕКТИВНОСТИ

Одной из главных характеристик классификатора является его эффективность. Самый очевидный способ оценивания эффективности — доля правильных ответов.

$$\text{Accuracy} = \frac{P}{N}, \tag{2.1}$$

где P — количество объектов, чей класс совпал с классом, определенным алгоритмом, а N — размер тестовой выборки.

Однако, у этой метрики есть некоторая проблема. Например, если алгоритм имеет Accuracy = 95 % — это хорошо? На самом деле не всегда. В случае если в тестовой выборке 95 % экземпляров одного класса и 5 % другого, то это может означать, что алгоритм все объекты записывает только в первый класс. Поэтому нам необходимы более точные инструменты оценки эффективности.

2.1. Точность и полнота

Точность (precision) и полнота (recall) являются метриками, которые используются при оценке большей части алгоритмов извлечения информации.

Для их определения введем некоторые понятия и составим матрицу ошибок (табл. 1):

- Если классификатор определил класс объекта 0 и его класс 0, назовем это **True Negative**, то есть классификатор правильно предсказал класс как негативный.
- Если классификатор определил класс объекта 0 и его класс 1, назовем это **False Negative**, то есть классификатор неправильно предсказал класс как негативный.
- Если классификатор определил класс объекта 1 и его класс 0, назовем это **False Positive**, то есть классификатор неправильно предсказал класс как позитивный.
- Если классификатор определил класс объекта 1 и его класс 1, назовем это **True Positive**, то есть классификатор правильно предсказал класс как позитивный.

Таблица 1. Матрица ошибок [12]

Количество наблюдений		Предсказанное состояние	
		Положительное предсказание	Отрицательное предсказание
Истинное состояние	Положительное состояние	Верно определенный положительный (TP)	Неверно определенный отрицательный (FN) (Ошибка II рода)
	Отрицательное состояние	Неверно определенный положительный (FP) (Ошибка I рода)	Верно определенный отрицательный (TN)

$$TP + FN = \text{Все положительные}, \tag{2.2}$$

$$TN + FP = \text{Все отрицательные}, \tag{2.3}$$

Точность системы (Precision) — это отношение количества верно определенных положительных объектов к общему количеству положительных объектов, которые классификатор отнес к этому классу, то есть точность определения положительных ответов. Чем ближе к единице precision, тем меньше неправильных определений классов, которые мы посчитали правильными [3].

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.4}$$

Полнота системы (Recall) — это отношение найденных классификатором объектов, принадлежащих классу, к общему количеству положительных объектов в тестовой выборке. Эта мера показывает, насколько хорошо мы в ходе классификации предсказали положительные ответы из всех положительных в выборке.

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.5}$$

2.2. *F*-мера

Важность точности и полноты зависит от конкретной задачи. Соответственно есть функция, которая делает предпочтения полноте или точности, так называемая F-мера:

$$F = (b^2 + 1)\frac{\text{Precision} \cdot \text{Recall}}{b^2 \cdot \text{Precision} + \text{Recall}}. \tag{2.6}$$

В тех случаях, когда неизвестно, какая из характеристик наиболее важна, берем $b = 1$ (F_1-мера). В этом случае F_1 стремится к 0, если точность или полнота стремятся к 0.

Точность и полнота регулируются при тренировке классификатора определенным параметром (predict_prob), называемым порогом. С помощью изменения порога можно увеличить Precision до заданного нами значения, вследствие этого Recall сильно упадет. Поэтому нам нужна независимая метрика, по которой мы можем оценить эффективность классификатора.

2.3. AUC

ROC (receiver operating characteristic, кривая ошибок) — кривая, позволяющая оценить качество бинарной классификации, задается параметрически:

$$x = \frac{FP}{TN + FP}, \tag{2.7}$$

$$y = \frac{TP}{TP + FN}. \tag{2.8}$$

x — False Positive Rate, то есть отношение неверно определенных в положительный класс ко всем элементам отрицательного класса.
y — True Positive Rate, то есть отношение верно определенных в положительный класс ко всем элементам положительного класса.

Фактически кривая показывает соотношение между долей верно определенных в положительный класс и долей ошибочно определенных в положительный класс.

Качество класификатора определяется площадью под кривой (AUC — area under curve, и является количественной характеристикой для ROC–кривой:

$$\text{AUC} = \int_0^1 \frac{TP}{TP+FN} d\frac{FP}{TN+FP}, \tag{2.9}$$

ROC-кривые являются примером применения AUC. В идеальном случае у нас FN и FP равны нулю (классификатор не допускает ошибок), следовательно, $TPR = \frac{0}{TN+0} = 0$ $FPR = \frac{TP}{TP+0} = 1$. В таком случае кривая принимает вид прямой $y = 1$ и площадь под такой прямой равна 1.

В худшем случае мы будем предсказывать класс с вероятностью 0,5. Тогда ROC-кривая примет вид прямой $y = x$ и площадь под ней будет равна 0,5.

3. МЕТОДЫ КЛАССИФИКАЦИИ

3.1. Метод k-ближайших соседей

Задана обучающая выборка $T = \{(x_1, y_1), ..., (x_m, y_m)\}, x_i \in X, y_i \in Y$, X — множество объектов, Y — множество классов, а m — мощность T [1]. На X введена функция расстояния $\rho(x, x')$. Чем больше значение этой функции, тем менее схожими являются объекты x и x'.

Для произвольного объекта $x \in X$ расположим объекты обучающей выборки x_i в порядке возрастания расстояний до x:

$$\rho(x, x_{1;x}) \leq \rho(x, x_{2;x}) \leq ... \leq \rho(x, x_{m;x}), \tag{3.1}$$

где $x_{i;x}$ — объект обучающей выборки, который является i-м соседом объекта x. Аналогичное обозначение вводится и для классов на i-м соседе $y_{i;x}$. Каждый $x \in X$ порождает свою нумерацию. Из этих упорядоченных элементов T выбираются первые k (задающийся параметр), а затем x присваивается тот класс, который преобладает среди этих k элементов.

В общем виде алгоритм k-ближайших соседей можно записать следующим образом:

$$a(x) = \underset{y \in Y}{\text{argmax}} \sum_{i=1}^{m} [y_{i;x} = y]\omega(i, x), \tag{3.2}$$

где $\omega(i; x)$ — заданная весовая функция, оценивающая насколько сильно i-й сосед влияет на объект x.

3.2. Наивный байесовский классификатор

Пусть у нас есть выборка объектов X, описываемая n характеристиками, и множество классов Y. Для каждого $x \in X$ выбираем такой класс, вероятность принадлежности к которому максимальна. Математически это записывается так:

$$c = \underset{y \in Y}{\text{argmax}} P(Y|x). \tag{3.3}$$

Такие вероятности проблематично вычислять, но мы можем перейти к условным вероятностям, используя формулу Байеса:

$$P(Y|x) = \frac{P(x|Y)P(Y)}{P(x)}. \tag{3.4}$$

Объекты из выборки $x \in X$ описываются n статистически независимыми признаками [8]. Используя это и то, что мы ищем максимум, а знаменатель — постоянная, зависящая от выборки, можем переписать формулу в следующем виде:

$$c = \underset{y \in Y}{\operatorname{argmax}} P(y) \prod_{i=1}^{n} P(x_i|Y). \tag{3.5}$$

Таким образом, необходимо вычислить $P(y)$ — вероятность того, что случайный объект будет принадлежать классу y, и $P(x_i|Y)$ — вероятность принадлежности объекта x_i к классу y. Вычисление этих параметров называется тренировкой классификатора.

3.3. Дерево принятия решений

Дерево принятия решений [13] — это алгоритм классификации, работающий над деревом (в нашем случае бинарным), в котором каждой внутренней вершине $v \in V$ приписан предикат $\beta_v : X \to \{0, 1\}$, каждой терминальной вершине $v \in V$ приписано имя класса $c_v \in Y$. При классификации объекта $x \in X$ он проходит по дереву путь от корня до некоторого листа.

Объект x доходит до вершины v тогда и только тогда, когда выполняется конъюнкция $K_v(x)$, составленная из всех предикатов, приписанных внутренним вершинам дерева на пути от корня до вершины v. Множества объектов $\Omega_v = \{x \in X : K_v(x) = 1\}$ попарно не пересекаются, а их объединение совпадает со всем пространством X.

Отсюда следует, что алгоритм классификации $a : X \to Y$, реализуемый бинарным решающим деревом, можно записать в виде простого голосования конъюнкций [11]:

$$a(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{c_v = y} K_v(x), \tag{3.6}$$

причем для любого $x \in X$ одно и только одно слагаемое во всех этих суммах равно единице.

Требование максимизации информативности конъюнкций $K_v(x)$ означает, что каждая из них должна выделять как можно больше обучающих объектов, допуская при этом как можно меньше ошибок. Число листьев в дереве должно быть как можно меньше, и они должны покрывать части выборки примерно одинаковой мощности.

3.4. Случайный лес

Метод случайного леса [4] основан на методе решающих деревьев. Случайный лес — это множество решающих деревьев, а класс объекта, проходящего классификацию, выбирается голосованием большинством.

- Сгенерируем случайную выборку S размера l по исходной обучающей выборке $D = \{x_i, y_i\}_{i=1}^{l}$.
- По выборке S индуцировать неусеченное дерево решений T_i с минимальным количеством наблюдений в терминальных вершинах равным n_{min}, рекурсивно следуя следующему подалгоритму:
  1) из исходного набора n признаков случайно выбрать p признаков,
  2) из p признаков выбрать признак, который обеспечивает наилучшее расщепление,

3) расщепить выборку, соответствующую обрабатываемой вершине, на две подвыборки.

- В результате получаем ансамбль деревьев решений $\{T_i\}_{i=1}^{B}$.
- Классификация новых наблюдений осуществляется следующим образом [16]: пусть $\hat{y}_i(x) \in \{y_1, y_2, ..., y_l\}$ — класс, предсказанный деревом решений T_i, то есть $T_i(x) = \hat{y}_i(x)$, тогда $\hat{y}_{rf}^{B}(x)$ — класс, наиболее часто встречающийся в множестве $\{\hat{y}_b(x)\}_{i=1}^{B}$.

3.5. Метод опорных векторов

Идея метода [5] состоит в том, чтобы найти разделющую объекты пространства $\mathbb{R}^n$ гиперплоскость, расстояние от которой до точек обоих классов максимально. Задана обучающая выборка $T = \{(x_1, y_1), ..., (x_m, y_m)\}$, $x_i \in \mathbb{R}^n$, $y_i \in \{0, 1\}$, а m — мощность T.

Метод опорных векторов строит классифицирующую функцию F в виде

$$F(x) = \text{sign}(\langle w, x\rangle + b), \tag{3.7}$$

где $\langle,\rangle$ — скалярное произведение, w — нормальный вектор к разделяющей гиперплоскости, b — вспомогательный параметр. Объекты, для которых $F(x) = 1$, попадают в один класс, если $F(x) = -1$ — во второй. Функция выбрана таким образом, потому что любая гиперплоскость может быть задана в виде $\langle w, x\rangle + b = 0$ для некоторых w и b.

Далее нам необходимо выбрать такие w и b, которые обеспечивают максимальное расстояние до каждого класса. Данное расстояние равно $\frac{1}{\|w\|}$, а проблема нахождения максимума $\frac{1}{\|w\|}$ эквивалентна проблеме нахождения минимума $\|w\|^2$, и мы можем записать задачу оптимизации:

$$\begin{cases} \text{argmin}_{w,b} ||w||^2, \\ y_i(\langle w, x_i\rangle + b) \geq 1, \ i = 1, ..., m, \end{cases} \tag{3.8}$$

которая является стандартной задачей квадратичного программирования и решается с помощью множителей Лагранжа.

Для нелинейного разделения классов следует вложить пространство признаков $\mathbb{R}^n$ в пространство со скалярным произведением H большей размерности с помощью отображения $\phi : \mathbb{R}^n \to H$. Тогда, рассматривая алгоритм опорных векторов для образов $\phi(x_i)$ обучающей выборки, разделяющую функцию будем искать в виде

$$F(x) = \langle \omega, \phi(x)\rangle + b, \tag{3.9}$$

$$\omega = \sum_{i=1}^{N} \lambda_i y_i \phi(x_i), \tag{3.10}$$

где коэффициенты λ_i зависят от y_i и от значения $(\phi(x_i), \phi(x_j))$.

$$K(x, y) = (\phi(x), \phi(y)), \tag{3.11}$$

Функция $K(x, y)$ называется ядром.

Наиболее распространенные ядра:

- Радиальное (Radial Basis Function — RBF):

$$k(x, x') = \exp(-\gamma|x - x'|^2). \tag{3.12}$$

- Полиномиальное:

$$k(x, x') = (x \cdot x')^d. \tag{3.13}$$

3.6. Многослойный перцептрон

Многослойный перцептрон [8] — нейронная сеть, состоящая из слоев, каждый из которых состоит из элементов — искусственных нейронов. Эти элементы бывают трех типов: сенсорные (входные, S), ассоциативные (обучаемые «скрытые» слои, A) и реагирующие (выходные, R). Искусственный нейрон — это элемент сети, имеющий несколько входов, каждый из которых имеет вес. Нейрон, получая сигнал (x — вектор входных сигналов в слой), умножает сигналы на веса (W — матрица весов между двумя рядом стоящими слоями) и суммирует получившиеся величины, после чего передает результат (s — вектор выходных сигналов слоя) другому нейрону или на выход сети.

$$s = f(x; W) = Wx. \tag{3.14}$$

Выходной сигнал проходит через функцию активации, которая приводит сигналы всех нейронов к одному удобному типу (например, помещает все значения в промежуток [0, 1]).

Обучение производится методом обратного распространения ошибки: после сравнения ответа перцептрона с правильным ответом вычисляется ошибка, которая передается обратно по всем слоям сети и изменяет веса связей. Чтобы этот метод мог использоваться, функция активации должна быть дифференцируема.

В некоторых случаях (включая перцептрон) используются такие функции активации как гиперболический тангенс и логистическая функция из-за простоты вычисления их производных.

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{3.15}$$

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)), \tag{3.16}$$

$$\tau(x) = \text{th}(\frac{x}{\alpha}) = \frac{e^{\frac{x}{\alpha}} - e^{-\frac{x}{\alpha}}}{e^{\frac{x}{\alpha}} + e^{-\frac{x}{\alpha}}}, \tag{3.17}$$

$$\tau'(x) = (1 + \tau(x)) \cdot (1 - \tau(x)). \tag{3.18}$$

3.7. Линейный дискриминантный анализ

Линейный дискриминантный анализ [8] — метод, применяемый для нахождения линейных комбинаций признаков, наилучшим образом разделяющих два или более класса объектов или событий. Полученная комбинация может быть использована в качестве линейного классификатора или для сокращения размерности пространства признаков перед последующей классификацией.

Задана обучающая выборка $T = \{(x_1, y_1), ..., (x_m, y_m)\}$, x_i — признаки объекта, $y_i \in \{0, 1\}$ — класс объекта, а m — мощность T. Задача классификации — построить

хороший прогноз класса y_i, имея только признаки объекта (для дальнейшего удобства $x = \vec{x}$).

При LDA предполагается, что функции совместной плотности распределения вероятностей $p(\vec{x}|y = 1)$ и $p(\vec{x}|y = 0)$ нормально распределены. В этих предположениях оптимальное байесовское решение — относить точки ко второму классу, если отношение правдоподобия ниже некоторого порогового значения F:

$$(\vec{x} - \vec{\mu}_0)^T \Sigma_0^{-1} (\vec{x} - \vec{\mu}_0) + \ln|\Sigma_0| - (\vec{x} - \vec{\mu}_1)^T \Sigma_1^{-1} (\vec{x} - \vec{\mu}_1) - \ln|\Sigma_1| \leq F, \tag{3.19}$$

где $\vec{\mu}_0, \vec{\mu}_1$ — средние плотностей, Σ_0, Σ_1 — параметры ковариации.

Далее предположим гомоскедастичность (то есть классы ковариации идентичны: $\Sigma_0 = \Sigma_1 = \Sigma$) и что ковариации имеют полный ранг. В этом случае несколько членов исключаются:

$$\vec{x}^T \Sigma_0^{-1} \vec{x} = \vec{x}^T \Sigma_1^{-1} \vec{x}, \tag{3.20}$$

$$\vec{x}^T \Sigma_i^{-1} \vec{\mu}_i = \vec{\mu}_i^T \Sigma_i^{-1} \vec{x}, \tag{3.21}$$

так как Σ_i — эрмитова, и описанный выше критерий решения становится пороговым значением для скалярного произведения $\langle \vec{w}, \vec{x} \rangle \leq f$ с некоторой пороговой константой f, где

$$\vec{w} = \Sigma^{-1} (\vec{\mu}_1 - \vec{\mu}_0) \tag{3.22}$$

$$f = \frac{1}{2} (F - \vec{x}_0^T \Sigma_0^{-1} \vec{\mu}_0 + \vec{x}_1^T \Sigma_1^{-1} \vec{\mu}_1) \tag{3.23}$$

Это означает, что критерий для вероятности принадлежности $\vec{x}$ классу y зависит только от линейной комбинации известных наблюдений.

3.8. CatBoost

CatBoost (CB) — это продвинутая библиотека с открытым исходным кодом градиентного бустинга на деревьях решений. Это преемник MatrixNet — алгоритма, применяемого для ранжирования и прогнозирования. CatBoost использует более универсальный алгоритм, поэтому подходит для решения и других задач. CatBoost гибок в представлении индексов категориальных столбцов, чтобы использовать прямое кодирование.

При сравнительном тестировании на популярных наборах данных CatBoost выигрывает у аналогов, также в библиотеке реализован эффективный метод кодирования, уменьшающий переобучение.

Преимущество CatBoost состоит в том, что он учитывает помимо числовых данных еще и нечисловые, то есть нечисловые данные можно использовать в первоначальном виде, не переводя их в цифры и не теряя тем самым суть параметров, влияющих на точность предсказания модели.

4. КРОСС-ВАЛИДАЦИЯ

Кросс-валидация [14] (иногда называется перекрестной проверкой) — техника валидации модели для проверки того, насколько хорошо применяемый статистический анализ способен работать на независимом наборе данных. Она используется в ситуациях,

где мы хотим получить предсказание и надо оценить, насколько модель способна работать на практике. Один цикл кросс-валидации включает в себя разбиение данных на тестовую и обучающую выборку, затем построение модели на обучающей выборке. Чтобы уменьшить разброс результатов, разные циклы кросс-валидации проводятся с помощью разных разбиений, а её результаты усредняются по всем циклам.

Этот метод важен для защиты модели от ошибок, навязанных данными, особенно когда получение новых данных невозможно.

Предположим, у нас есть модель с некоторым количеством параметров и тестовая с обучающей выборкой. Процесс тренировки подгоняет параметры модели так, чтобы они наилучшим образом подходили к тестовой выборке. Теперь, если мы проверим модель на тестовой выборке, скорее всего, мы обнаружим результаты хуже, чем на обучающей. Это называется переобучением (overfitting), которое обычно встречается в ситуациях, когда размер обучающей выборки невелик или в модели слишком много параметров. Кросс-валидация помогает оценить способность модели работать на гипотетическом тестовом наборе данных, когда в реальности его получить нельзя.

4.1. Кросс-валидация по *k* блокам

В этом случае исходный набор данных разделяется на k одинаковых по размеру блоков. Затем выбирается один блок, который будет использоваться как тестовая выборка, оставшиеся идут в обучающую. Процесс повторяется, пока каждый блок не будет использован в качестве тестового. Результаты по всем k блокам усредняются (или комбинируются более специфическим образом) и дают одну оценку. Схема работы кросс-валидации представлена на рисунке 1.

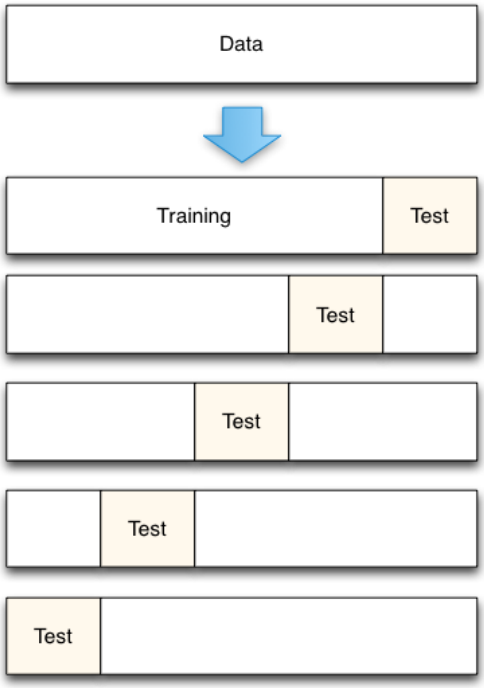

Рис. 1. Схема работы кросс-валидации [17]

4.2. Валидация случайным семплированием

Этот метод случайным образом разбивает набор данных на обучающую и тестовую выборку. Для каждого разбиения модель тренируется, затем происходит оценка точности предсказания. Результаты комбинируются по всем разбиениям. Преимущество этого метода над кросс-валидацией по k блокам в том, что пропорции обучающей и тестовой выборки не зависят от числа блоков. Недостаток же в том, что некоторые элементы могут ни разу не попасть в обучающую выборку, а другие могут попасть туда несколько раз. Кроме того, в случае других попыток анализа повторить обучение той же модели не предоставляется возможным, так как разбиения генерируются случайным образом.

4.3. Поэлементная кросс-валидация

То же самое, что и кросс-валидация по k блокам, но в качестве тестовой выборки используется один элемент, то есть k равно количеству наблюдений.

5. ПОДГОТОВКА ДАННЫХ

5.1. Бэггинг (Bagging)

Бэггинг на подпространствах — схожий метод ансамблирования, отличающийся от описанного выше тем, что модели обучаются независимо не на разных объектах, а на разных признаках. Иначе говоря, каждая модель знает про все объекты, но только лишь про свою часть их признаков.

Бэггинг на подпространствах — схожий метод ансаблирования, отличающийся от описанного выше тем, что модели обучаются независимо не на разных объектах, а на разных признаках. Иначе говоря, каждая модель знает про все объекты, но только лишь про свою часть их признаков.

5.2. Бустинг (Boosting)

Основной целью бустинга [19] является повышение эффективности классификации за счет сочетания решения нескольких моделей. Бустинг успешно применяется к прогнозированию оттока клиентов в розничной торговле и телекоммуникационных компаниях.

Имеются множества X и Y, дополнительно вводится множество R, называемое пространством оценок. Рассматриваются алгоритмы, имеющие вид суперпозиции

$$a(x) = C(b(x)), \tag{5.1}$$

где функция $b : X \to \mathbb{R}$ называется алгоритмическим оператором, функция $C : \mathbb{R} \to Y$ — решающим правилом. Сначала вычисляются оценки принадлежности объекта классам, затем решающее правило переводит эти оценки в номер класса. Значение оценки характеризует степень уверенности классификации. Алгоритмическая композиция — алгоритм $a : X \to Y$ вида

$$a(x) = C(F(b_1(x), ..., b_T(x))),\ x \in X, \tag{5.2}$$

составленный из алгоритмических операторов $b_t : X \to R,\ t = 1, ..., T$, корректирующей операции $F : R^T \to R$ и решающего правила $C : R \to Y$. В нашем случае рассматривается бинарная задача классификации: $Y = \{-1, +1\}$. Решающее правило фиксировано:

$$C(b) = \text{sign}(b). \tag{5.3}$$

Искомая алгоритмическая композиция имеет вид:

$$a(x) = C(F(b_1(x), ..., b_T(x))) = sign(\sum_{t=1}^{T} a_t b_t(x)),\ x \in X \tag{5.4}$$

После того, как веса всех объектов были определены, происходит их пересчет, который выглядит следующим образом: если b_t правильно определяет класс объекта, то его вес уменьшается в e^{at} раз, иначе вес увеличивается во столько же раз. Это позволяет задать больший вес тем объектам, которые чаще остальных вызывали трудности.

В результате мы получаем линейную комбинацию довольно слабых классификаторов, каждый из которых хорошо обрабатывает тот или иной трудный случай. За счет этого получаем хороший результат.

5.3. Решетчатый поиск (Grid Search — GS)

С помощью Grid Search можно осуществлять подборку параметров. Обязательными для передачи решетчатому поиску являются такие параметры, как модель классификатора, стратегия кросс валидации, метрика, по которой оценивается эффективность модели (например AUC). Пусть у нас n параметров классификатора, которые варьируются в определенных границах, получается, что мы передаем n-мерный объект, в границах которого для каждой точки вычисляем значения модели на данных показателях, как итог — из всех возможных выбирается лучший набор параметров по метрике.

Однако существует очевидная проблема — многократное увеличение времени работы алгоритма. Причем, если для некоторых классификаторов это не критично (от пары секунд до минуты), то, например, для SVM, необходимое время может составлять часы (есть методы решения этой проблемы, например случайный поиск по сетке для постепенного сужения области максимума). Рассмотрим алгоритм работы GS на примере SVM.

В дополнение к выбору подмножества компонентов и настройке ядра (3.11) для увеличения точности классификации SVM [5] в этой работе мы выбрали RBF (3.12). Чтобы получить лучшую производительность, нужно оптимизировать два параметра RBF: C и γ, где C — параметр штрафа, γ — параметр ядра. Оба параметра играют решающую роль в работе SVM. Применяется GS на C и γ с n-кратной кросс-валидацией для оптимизации этих параметров. Пара (C, γ), полученная из самого большого результата по метрике AUC, используется для дальнейшего анализа.

Неправильный выбор этих параметров может быть непродуктивным. Предварительно невозможно узнать, какая комбинация (C, γ) приведет к лучшей производительности при проверке тренировочных данных для предсказания. Необходимо выполнить некоторую процедуру выбора параметров.

Выполняя n-кратную кросс-валидацию, тренировочные данные разделяются наn подмножеств равного размера. Одна часть не учитывается для валидации, а остальные $(n-1)$-части используются для обучения. Чтобы определить, какая пара параметров дает лучший результат, эту процедуру стоит повторить с несколькими парами (C, γ).

Все комбинации проверяются, и остаются две пары: 1) с лучшей точностью после кросс-валидации; 2) с лучшим AUC после кросс-валидации. Эта пара выбрана по той причине, что AUC учитывает чувствительность и специфичность в качестве отдельных показателей производительности метрик. После получения этих пар оптимальных параметров данные снова тренируются. Оба классификатора используются для предсказания в дальнейшем.

5.4. Метод главных компонент (Principal Component Analysis — PCA)

Метод главных компонент [9] предназначен для поиска и выявления параметров, имеющих линейную зависимость между собой, а также выделения различающихся компонент с целью уменьшения размерности данных.

Обозначим в исходной p-мерной системе координат переменные как $x_1, ..., x_p$. Каждая переменная x_i является одномерным вектором размера k, где k — количество выборок в наборе данных.

Метод главных компонент можно рассматривать как последовательность задач оптимизации:

$$y_i = \underset{\|y_i\|}{\operatorname{argmin}} \left(\sum_{i=1}^{m} \|x_i - y_i(y_i, x_i)\|^2 \right). \tag{5.5}$$

PCA преобразует набор данных в искусственные параметры, которые покрывают максимальную дисперсию исходных. Суть в том, чтобы объяснить корреляцию между непрерывными независимыми переменными, используя линейные комбинации новых, называемых главными компонентами. i-ая главная компонента определяется следующим образом:

$$y_i = e_i^{'} x = \sum_{j=1}^{p} e_{ij} x_j, \tag{5.6}$$

где $i = 1, ..., k$ и $e_i^{'}$ — это транспонирование i-го собственного вектора. Эта система поворачивается в направлении максимальной дисперсии, образуя новую систему координат, представленную главными компонентами. Получаем n главных компонент, которые объясняют общую вариацию исходного набора данных, все приведенные компоненты являются независимыми, $n \leq p$.

i-ая главная компонента не зависит от всех предыдущих компонент и максимизирует дисперсию:

$$\operatorname{var}(y_i) = e_i^{'} \rho e_i. \tag{5.7}$$

Дисперсия, объясняемая i-й основной компонентой относительно полной дисперсии, вычисляется с использованием отношения $\frac{\lambda_i}{p}$, где λ_i — i-ое собственное значение.

Новый набор данных используется далее для обучения.

6. СРАВНЕНИЕ РАЗЛИЧНЫХ НАБОРОВ ДАННЫХ

Характеристики ноутбука:

- ЦПУ: Intel(R) Core(TM) i3-3127U 1.80Гц
- ОЗУ: 4,00 Гб

В ходе исследования нами были проанализированы четыре базы данных (табл. 2): три датасета находятся в открытом доступе [15, 18, 20] и один датасет предоставлен российской мобильной компанией. В базах данных хранится вся доступная информация о клиенте (активность, транзакции, личная информация и т. д.), то есть каждый абонент рассматривается как совокупность этих характеристик (features).

В первом открытом датасете (Д1) [15] имеется 15 числовых параметров и 5 категориальных (один из которых — ушел абонент или нет). Данная выборка обладает несбалансированными классами.

Таблица 2. Результаты для 4 датасетов (%)

	AUC (%)				Время обучения (с)				Accuracy (%)				F_1-мера (%)			
	Д1	Д2	Д3	Д4	Д1	Д2	Д3	Д4	Д1	Д2	Д3	Д4	Д1	Д2	Д3	Д4
NN	90	86	91	96	0.95	3.56	2.98	97	91	91	91	89	90	90	88	90
NN (Bag)	91	86	91	96	25.94	34.41	23.18	1128	91	91	91	90	90	90	89	90
NN (PCA)	90	88	89	95	2.64	3.56	1.93	108	91	92	91	88	91	91	91	87
NN (GS)	86	87	91	96	–	–	–	–	92	92	88	90	90	90	84	88
NN (Boost)	92	88	92	96	2.35	10.17	10.15	194	93	92	93	90	91	91	90	91
KNN	86	86	86	89	0.18	0.07	0.25	93	89	89	88	80	87	87	85	81
KNN (Bag)	86	86	86	90	0.46	0.15	0.33	378	87	89	87	80	87	87	85	81
KNN (PCA)	86	88	88	86	0.24	0.06	0.21	67	90	90	89	81	88	89	86	82
KNN (GS)	86	86	86	89	–	–	–	–	89	89	88	87	87	87	84	81
KNN (Boost)	88	87	89	93	0.47	0.34	1.13	146	90	90	90	86	88	88	88	85
DT	92	88	91	89	0.22	0.11	0.3	9	91	86	90	75	91	86	92	75
DT (Bag)	92	90	90	87	2.3	1.75	1.5	40	96	93	95	70	95	92	94	80
DT (PCA)	87	85	86	86	0.36	0.16	0.26	16	85	84	85	75	86	83	86	73
DT (GS)	94	90	93	91	–	–	–	–	94	90	94	85	94	91	94	87
DT (Boost)	92	89	93	94	2.44	0.18	1.25	44	96	93	95	87	91	92	95	83
SVM	88	88	87	95	1.37	2.87	1.6	4980	90	90	90	89	90	89	89	90
SVM (Bag)	88	87	87	95	10.23	12.18	6.3	22139	90	90	89	89	89	89	88	89
SVM (PCA)	89	88	89	92	2.07	2.52	1.42	5115	91	91	90	86	91	90	90	86
SVM (GS)	91	88	91	93	–	–	–	–	91	90	90	89	90	89	89	90
SVM (Boost)	90	89	86	95	146	170	76	7326	85	87	87	90	81	81	81	90
NB	86	86	85	90	0.18	0.07	0.26	4	87	87	87	83	81	81	81	83
NB (Bag)	86	86	85	90	0.38	0.11	0.27	5	87	87	87	83	81	81	81	83
NB (PCA)	86	86	85	84	0.24	0.08	0.2	3	87	87	87	72	80	81	81	72
NB (GS)	86	86	85	82	–	–	–	–	87	87	87	79	80	81	81	83
NB (Boost)	86	86	85	92	0.48	0.44	0.37	7	87	87	87	85	81	81	81	84
RF	92	87	88	86	0.23	0.17	0.36	8	95	92	95	73	94	91	94	72
RF (Bag)	92	87	93	88	1.37	0.85	1.01	32	95	93	95	73	95	91	95	73
RF (PCA)	87	86	86	88	0.45	0.22	0.35	17	90	91	89	80	89	89	88	81
RF (GS)	93	87	92	92	–	–	–	–	95	92	95	89	94	91	94	90
RF (Boost)	88	88	90	88	3.53	2.62	2.84	109	92	92	92	83	90	91	91	83
LDA	86	86	86	93	0.25	0.09	0.26	5	87	87	86	87	85	81	84	87
LDA (Bag)	86	86	86	93	0.42	0.14	0.33	18	87	87	86	87	85	81	85	87
LDA (PCA)	86	86	86	85	0.25	0.05	0.17	4	86	87	86	75	84	83	84	73
LDA (GS)	86	86	85	85	–	–	–	–	87	87	86	84	85	81	84	87
LDA (Boost)	89	89	87	94	0.73	0.42	0.67	14	88	87	88	89	87	83	86	88
CB	93	90	94	95	0.75	0.72	0.63	7	96	93	96	90	95	92	95	89
CB (Bag)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CB (PCA)	88	86	89	92	0.65	0.47	0.53	7	90	92	90	85	91	92	90	86
CB (GS)	93	89	93	94	–	–	–	–	95	93	96	91	96	93	97	90
CB (Boost)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Второй датасет (Д2) [18] состоит из информации о 5000 клиентах и включает в себя независимые переменные, такие как длина учетной записи, количество сообщений голосовой почты, общий дневной сбор, общий вечерний сбор, общий ночной сбор, общий международный сбор и количество звонков в службу поддержки. Зависимая переменная в наборе данных показывает, ушел абонент или нет, обозначена 1 для «да» и 0 для «нет».

Третий датасет (Д3) [20] состоит из информации о 3300 клиентов, включая в себя 3 номинативных переменных и 17 количественных переменных. Этот набор данных также предоставляет полную информацию об абоненте телекоммуникационной компании, с помощью которой проводится анализ оттока. В конце имеется зависимая переменная, обозначающая ушел клиент или нет, 1 или 0 соответственно.

Четвертый датасет (Д4): для обучения моделей использовались реальные данные пользователей одного из крупнейших в России операторов сотовой связи. Были предоставлены актуальные данные 130000 клиентов. Для каждого абонента имелась информация о последних 4 месяцах его активности.

Также для каждого клиента было известно, ушел ли он за отчетный период от оператора. Таким образом, в наличии были размеченные данные достаточного размера для обучения моделей.

Данные были агрегированы по следующим категориям:

- Минуты и стоимость входящих вызовов.
- Минуты и стоимость исходящих вызовов.
- Количество и стоимость исходящих СМС.
- Количество входящих СМС.
- Количество трафика мобильного интернета и его стоимость.
- Информация об обращениях клиента в службу поддержки.
- Тарифный план клиента.
- Личные данные.

Часть параметров, зависящая от местоположения (в родном субъекте РФ, в стране, в роуминге), была разбита на отдельные составляющие.

Перед выводом результата введем несколько пояснений:

- Поскольку решетчатый поиск (GS) — это метод поиска параметров, дающих лучший результат по метрике Accuracy, то в данном случае оценка времени (обучения) не является параметром, по которому приемлемо оценивать успешность поиска.
- Сочетание бэггинга с бустингом не реализуемо, так как бэггинг CatBoost — это бэггинг градиентного бустинга на решающих деревьях, а бэггинг основан на параллельной работе классификаторов в отличии от бустинга.

6.1. Сравнение с результатами других статей

Для выявления лучшего алгоритма сравним полученные результаты с результатами из аналогичных исследований, выполненных на основе анализа открытых баз данных для некоторых из методов, разобранных нами выше. Целью подобного сравнения является показать, что CatBoost дает наилучшие результаты не только между методами, реализованными в нашей работе, но и в сравнении по метрикам с другими исследованиями.

Сравнение по Accuracy:

- В первом датасете лучший результат показали CatBoost, Boosting и Bagging DT — 96 %. В статье [15] результат RF — 94 %.
- Во втором датасете лучший результат показали CatBoost, Grid Search CB, Bagging RF, Boosting и Bagging DT — 96 %. В статье [18] результат LDA — 74 %.
- В третьем датасете лучший результат показали CatBoost и Grid Search CB — 96 %. В статье [20] результат DT — 95 %, по AUC – 90 %.
- В четвертом датасете лучший результат показал Grid Search CB — 91 %.

Сравнение результатов по метрике Accuracy выявило незначительное превосходство у метода CatBoost на каждом датасете. Большое значение на AUC оказывает Grid Search, как видно из представленных результатов — решетчатый поиск приводит к улучшению AUC.

Сравнение по AUC:

- В первом датасете лучший результат показал Grid Search DT — 94 %, а также CatBoost, Grid Search CB и Grid Search RF — 96 %.
- Во втором датасете лучший результат показали CatBoost, Grid Search DT и Bagging DT — 90 %.
- В третьем датасете лучший результат показал CatBoost — 94 %, а также Grid Search CB, Bagging RF, Boosting DT и Grid Search DT — 93 %. В статье [20] результат DT — 90 %.
- В четвертом датасете лучший результат показали NN, Bagging NN, Grid Search NN и Boosting NN — 96 %, а также PCA NN, SVM, Bagging SVM, Boosting SVM и CatBoost — 95 %.

Сравнение результатов по метрике AUC выявило прекрасные результаты, немного уступающие в первом и четвертом датасете.

Сравнение по F_1-мере:

- В первом датасете лучший результат показал Grid Search CB — 96 %, а также CatBoost, Bagging RF и Bagging DT — 95 %.
- Во втором датасете лучший результат показал Grid Search CatBoost — 93 %, а также CatBoost, PCA CB, Boosting DT и Bagging DT — 92 %.
- Во третьем датасете лучший результат показал Grid Search CatBoost — 97 %, а также CatBoost, Bagging RF и Boosting DT — 95 %.
- В четвертом датасете лучший результат показал Boosting NN — 91 %, а также Grid Search CB, Grid Search RF, Boosting SVM, Grid Search SVM, SVM, NN и Bagging NN — 90 %.

Сравнение результатов по F_1-мере также выявляет лучшие показатели у CatBoost в совокупности с решетчатым поиском.

7. ЗАКЛЮЧЕНИЕ

Исходя из результатов на датасетах, находящихся в открытом доступе, а также датасете российского мобильного оператора, основанном на «свежих» реальных данных об абонентах, мы получаем, что CatBoost имеет лучшие результаты AUC и Accuracy по сравнению с остальными методами классификации: DT, NN, KNN, LDA, SVM, RF, NB.

Время при использовании PCA ощутимо уменьшается на фоне почти не изменяющихся AUC и Accuracy, что безусловно является плюсом, так как с сокращением размерности данных происходит минимальная потеря их качества. Применение решетчатого поиска зачастую положительно сказывается на AUC, но в разы увеличивает время, затрачиваемое на получение набора параметров с лучшей оценкой, так как происходит перебор всех значений в заданной области. Идентичные результаты AUC, Accuracy и F_1-меры при использовании бэггинга не дают должного эффекта вкупе с многократно возрастающим временем обучения. На большинстве методов наблюдается стабильное улучшение оценок при использовании бустинга, но почти не достижимыми остаются результаты CatBoost.

В рамках исследования, целью которого являлась демонстрация превосходства метода CatBoost в мерах эффективности классификаторов, проведенные работы имели, помимо теоретической ценности, еще и практическую. Применение алгоритмов к реальным данным активизирует впоследствии действия компании, направленные на удержание клиентов. Успешно реализован прототип рабочей программы, используемой российской мобильной компанией для предсказания оттока абонентов.

Список литературы

1. *Гришанов К. М., Белов Ю. С.* Метод классификации K-NN и его применение в распознавании символов / Фундаментальные проблемы науки. Сборник статей Международной научно-практической конференции 15 мая 2016 г. Ч. 3. Тюмень: НИЦ Аэтерна, 2016. С. 30–33.
2. *Карякина А. А., Мельников А. В.* Сравнение моделей прогнозирования оттока клиентов интернет-провайдеров // Машинное обучение и анализ данных, 2017. Том 3, № 4. С. 250–256.
3. *Пономарёв А. А.* Сегментация пользователей мобильных операторов с помощью моделей Больших Данных. СПбГУ, 2018. URL: https://dspace.spbu.ru/bitstream/11701/11992/1/vkr.docx (дата обращения 20.09.2018).
4. *Чистяков С. П.* Случайные леса: обзор // Труды Карельского научного центра РАН. 2013. № 1. С. 117–136.
5. *Akay M. F.* Support vector machines combined with feature selection for breast cancer diagnosis // Expert Systems with Applications. 2009. Vol. 36(2). doi:10.1.1.473.6145
6. *Albadawi S., Latif K., Kharbat F.* Telecom Churn Prediction Model Using Data Mining Techniques [Bahria University Journal of Information & Communication Technologies], 2017. Vol 10. № Special Issue. P. 8–14.
7. *Dziaugyte S., Mzyk M.* Churn analysis — machine learning. Bloomington, 2016.
8. *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning: Data Mining, Inference, and Prediction. N.Y.: Springer, 2009.
9. *Jolliffe I.T.* Principal Component Analysis / Springer Series in Statistics. N.Y.: Springer, 2002. doi:10.1007/b98835
10. *Karthik Subramanya.* Enhanced feature mining and classifier models to predict customer churn for an e-retailer / A thesis for the degree of MS. Iowa State University. Ames, 2016.
11. *Keramati A., Jafari-Marandi R., Aliannejadi M., Ahmadianc I., Mozaffari M., Abbasia U.* Improved churn prediction in telecommunication industry using data mining techniques (Applied Soft Computing), 2014. Vol. 24. P. 994–1012.
12. *Kriti M.* A Machine Learning Approach for Churn Prediction in Telecommunication // International Conference on Energy, Communication, Data Analytics and Soft Computing. Chennai, India, 2017.
13. *Lomax S.,Vadera S.* Case Studies in Applying Data Mining for Churn Analysis [International Journal of Conceptual Structures and Smart Applications], 2017. № 5 (2). P. 22–33.
14. *Mullin M., Sukthankar R.* Complete cross-validation for nearest neighbor classifiers // Proceedings of International Conference on Machine Learning. San Francisco, CA, 2000.
15. *Oates S.* Churn Analysis. Sydney, 2018.
16. *Prashanth R., Deepak K.* High Accuracy Predictive Modelling for Customer Churn Prediction in Telecom Industry // Machine Learning and Data Mining in Pattern Recognition. N.Y., 2017. doi:10.1007/978-3-319-62416-7_28
17. *Scott F.-R.* Accurately Measuring Model Prediction Error, 2012. URL: http://scott.fortmann-roe.com/docs/MeasuringError.html
18. *Sowmya V.* Using Linear Discriminant Analysis to Predict Customer, 2018. URL: https://www.datascience.com/blog/predicting-customer-churn-with-a-discriminant-analysis
19. *Viola P., Jones M.* Rapid Object Detection using a Boosted Cascade of Simple Features, in *Accepted Conference on Computer Vision and Pattern Recognition*, 2001.
20. *Luqi Yao.* Customer Churn Prediction, USA, 2016. URL: http://rpubs.com/LuqiYao/churn (дата обращения 20.09.2018).

Поступила в редакцию 16.08.2018, окончательный вариант — 20.09.2018.

Computer tools in education, 2018
№ 5: 5–23
http://ipo.spb.ru/journal
doi:10.32603/2071-2340-2018-5-5-23

FORECASTING SUBSCRIBER CHURN: COMPARISON OF MACHINE LEARNING METHODS

Arzamastsev S. A., Bgatov M. V., Kartysheva E. N., Derkunskii V. A., Semenchikov D. N.

Saint Petersburg State University, Russia, Saint Petersburg, Russia

Abstract

In order to remain competitive today in the telecommunications business, it is necessary to identify customers who are dissatisfied with the services provided. Therefore, forecasting subscriber churn has become an essential issue in this area.

This article overviews different machine learning techniques including Decision Trees (DT), Naive Bayes Classifier (NB), Random Forest (RF), Artificial Neural Network (NN), K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Support Vector Machine (SVM) and their ensembles (bagging and boosting) in order to demonstrate the superiority of the CatBoost technology in gaging the effectiveness of classifiers. To achieve the goal, data was classified and the specific advantages, when compared to others, of the CatBoost method were revealed based on obtained results.

For the study, we analyzed four databases: 3 datasets are in open access and 1 dataset was provided by a Russian mobile company. Often, the dimension of these databases is high, which leads to a number of problems (including class imbalances, parameter correlations), which are solved by employing the dimensionality reduction method: Principal Component Analysis (PCA).

The results obtained are compared with each other as well as with the results presented by other researchers based on open databases. The effectiveness of classifiers is evaluated using measures such as the area under the curve (AUC), accuracy, F_1 -measure, and time.

Keywords: *data mining, machine learning, data balancing, data preparation, ensemble.*

Citation: S. A. Arzamastsev, M. V. Bgatov, E. N. Kartysheva, V. A. Derkunskii, and D. N. Semenchikov, “Churn Prediction: Comparison of Machine Learning Techniques”, *Computer tools in education*, no. 5, pp. 5–23, 2018 (in Russian). doi:10.32603/2071-2340-2018-5-5-23

Received 16.08.2018, the final version — 20.09.2018.

Arzamastsev Svyatoslav Aleksandrovich, student of SPbSU, Mathematics & Mechanics Faculty, Department of Statistical Modelling; 198504 Saint Petersburg, Stary Peterhof, Universitetsky pr., 28, cab. 4399, st037590@student.spbu.ru

Bgatov Mikhail Vladimirovich, student of SPbSU, Mathematics & Mechanics Faculty, st047070@student.spbu.ru

Kartysheva Elena Nikolavevna, student of SPbSU, Mathematics & Mechanics Faculty, st048188@student.spbu.ru

Derkunskii Viktor Arturovich, student of SPbSU, Mathematics & Mechanics Faculty, st047728@student.spbu.ru
Semenchikov Dmitrii Nikolaevich, postgraduate student of SPbSU, Faculty of Applied Mathematics and Control Processes, st016311@student.spbu.ru

Арзамасцев Святослав Александрович,
студент кафедры статистического моделирования математико-механического факультета СПбГУ;
198504 Санкт-Петербург, Петергоф, Университетский пр., 28, каб. 4399,
st037590@student.spbu.ru

Бгатов Михаил Владимирович,
студент математико-механического факультета СПбГУ,
st047070@student.spbu.ru

Картышева Елена Николаевна,
студентка математико-механического факультета СПбГУ,
st048188@student.spbu.ru

Деркунский Виктор Артурович,
студент математико-механического факультета СПбГУ,
st047728@student.spbu.ru

Семенчиков Дмитрий Николаевич,
аспирант факультета прикладной математики — процессов управления СПбГУ,
st016311@student.spbu.ru

© Наши авторы, 2018.
Our authors, 2018.