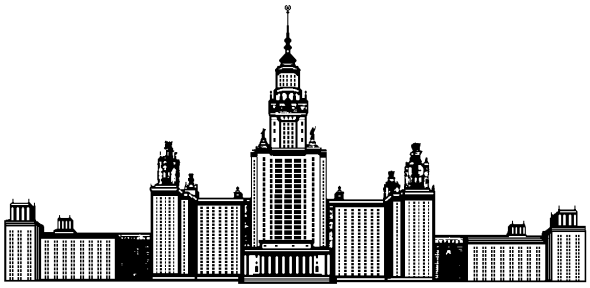

Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Кузнецов Михаил Константинович

Автоматическая генерация признаков на табличных данных

Automatic feature generation for tabular data

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
д.ф.-м.н., профессор
А. Г. Дьяконов

Москва, 2022

Содержание

§1. Введение

Сегодня машинное обучение применяется в большом количестве областей. В ряде задач нет много данных, поэтому нейронные сети не могут быть в полной мере применены. Остаются алгоритмы ml, которым нужны хорошие признаки. Чаще всего процесс решения конкретной задачи выглядит следующим образом: сбор данных, их предобработка, генерирование новых признаков, выбор критерия качества, выбор модели и ее обучение. В зависимости от того, насколько простая будет связь между признаками и целевой меткой, настолько модель будет быстрее обучаться и иметь меньшую сложность.

Из-за экспоненциального роста числа комбинаций признаков и их трансформаций задача построения полезных признаков трудозатратна. Появляется естественное желание автоматизировать этот процесс. Одна из ключевых подзадач в области автоматического машинного обучения (AutoML) [1] — умная генерация признаков.

В данной работе описываются методы автоматической генерации признаков на табличных данных, проводятся эксперименты как на реальных, так и на искусственных данных.

§2. Обзор литературы

Основное деление методов автоматической генерации признаков:

- генерация-отбор или итеративный (expand reducion):
  - все сразу генерируем признаки [2], [3], [4],
  - только перспективные [5].
- генетический [6], [7].
- с помощью дерева [8], [9], [10], [11].
- с помощью нейросетей [12], [13], [14].

Деление трансформаций, используемых для получения новых признаков:

1. Унарные:
   - применение нелинейных функций: exp(x), log(1 + x).
2. Бинарные:
   - +, -, /, *,
   - агрегация (groupby + func).
3. Многоуровневые:
   - кодирование категориальных признаков (целевое кодирование (TE), количественное кодирование (count encoding)),
   - методы снижения размерности:
     - автокодировщиком (AutoEncoder),
     - методом главных компонент (PCA) [15],
     - путем максимизации непараметрической взаимной информации (MMI) [16].
   - knn признаки в задаче без учителя (расстояния между наблюдениями, признаки ближайших наблюдений),
   - предсказание одних признаков по другим,
   - показатель качества / функция потерь восстановленных данных с помощью других моделей (плотность, mse),
   - кластеризация,
   - взаимодействия категориальных признаков:

– декартово произведение (crossover),
– M-of-N - Истина, когда хотя бы M из N условий верны,
– ДНФ, оставляем важные исходя из коэффициентов Фурье [17].

Большинство методов, основанных на генерации-отборе, имеют встроенный отборщик признаков, качестве которого выступают:

- взаимная информация,
- корреляция и корреляция расстояния (distance correlation) [18],
- веса линейной модели,
- оценка важности мета-моделью.

Например, в ExploreKit [2] авторы обучают мета-модель на большом объеме данных и предсказывают вероятность того, что данная трансформация признака даст прирост в качестве. В данном методе итоговое качество признака меряется как разница качества на исходном датасете с признаком и без него. Few [6] в отличии от метода главных компонент имеет сложность, зависящую линейно от количества объектов в выборке. Это позволяет применять его на больших данных.

FCDRGP [19] полезен для бустинга, так как он подбирает признаки исходя из показателя качества, основанного на разделении классов, так называемого fitness function. Суть в том, что если мы засэмлируем часть объектов из класса “c” и посмотрим какие объекты лежат недалеко от центра выборки, то в «хорошем» случае мы увидим объекты того же класса “c”, то есть энтропия должная быть низкая. Остаются признаки с относительно большим fitness function.

Преимущество SAFE [2] подхода перед итеративными заключается в сложности, зависящей от квадрата логарифма числа признаков, а не от квадрата числа признаков.

§3. Постановка задачи

Пусть выборка обозначается как $D = (x_i, y_i)_{i=1}^n$, где $(x_i, y_i) = ((x_{i1}, x_{i2}, ..., x_{ip}), y_i)$. $X = (x_i)_{i=1}^n$, $Y = (y_i)_{i=1}^n$. Допустим мы обучаем модель $\mathbf{a}$. Пара $(D, \mathbf{a})$ формирует конкретную ситуацию. Обозначим за $\mathcal{S}$ набор из всевозможных таких пар.

Тогда задача в общем случае выглядит следующим образом: необходимо задать функцию $\phi : \mathcal{S} \to \mathbb{R}^{n \times d}$, которая возвращает трансформированные данные.

Естественно, интерес заключается в нахождении новых признаков, которые:

- улучшают показатель качества решения итоговой задачи,
- быстро считаются,
- полезны для моделей машинного обучения и задач, отличных от пары $(D, \mathbf{a})$.

В частном случа есть набор трансформаций $\mathrm{tr}_1, \mathrm{tr}_2, ..., \mathrm{tr}_\mathrm{N}$ над признаками. Последовательность трансформаций $\mathrm{tr}_{\mathrm{i}_1}, \mathrm{tr}_{\mathrm{i}_2}, ..., \mathrm{tr}_{\mathrm{i}_\mathrm{m}}$ над исходными признаками называется формулой. Задача состоит в нахождении оптимального с точки зрения показателя качества модели $\mathbf{a}$ конечного набора формул, размер которого ограничен.

§4. Методы решения

4.1. Autofeat

Авторы данной статьи [3] сначала применяют классический подход: генерация унарных трансформаций признаков и затем использование бинарных трансформаций. Например, в качестве унарной трансформации взяли sin(x), а в качестве бинарной “+”. Для отбора признаков применялись следующие методы:

1. Использование веса модели линейной регрессии в качестве важности признака: чем меньше модуль веса, тем меньше важность.
2. Корреляция с другими признаками должна быть меньше определенного порога.
3. Для поиска полезных признаков брались подмножества трансформированных признаков, в которые добавлялись зашумленные важные признаки с шага 1. Если вес какого-то признака оказался больше веса зашумленного, такой признак считался «хорошим».
4. Отбор на основе длины формулы. Чем меньше, тем лучше.

Отбор происходит итеративно: на каждом шаге генерируются и отсеиваются ненужные признаки.

4.2. Neural Feature Search (NFS)

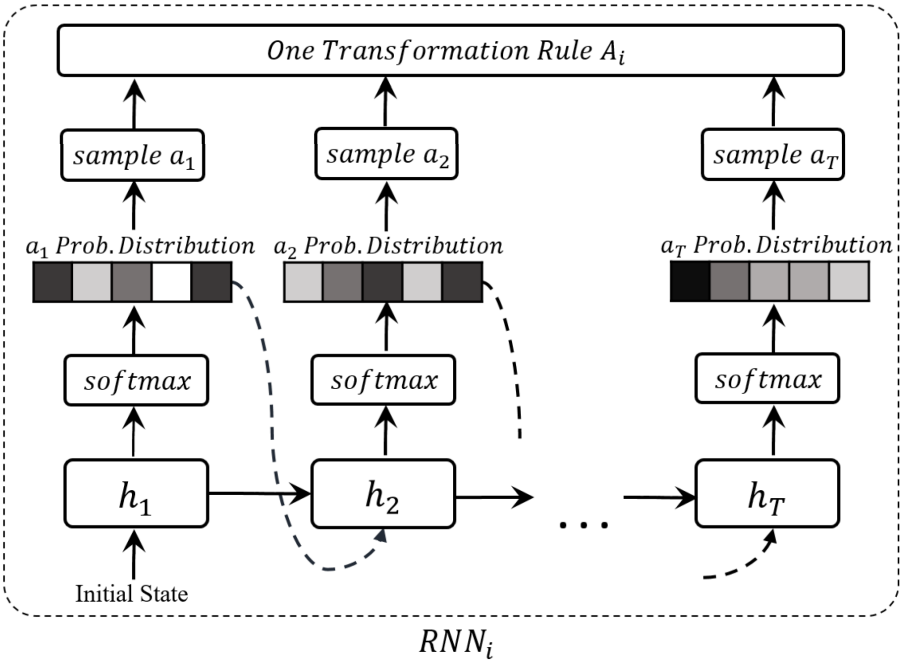

Рис. 1: Схема работы нейронной сети для генерации трансформаций в методе nfs [12]

Представим, что для каждого признака у нас есть модель, которая предсказывает какие трансформации стоит сделать, то есть в конце модели будет стоять софтмакс (softmax) преобразование, дающее нам вероятности полезности трансформаций. Трансформации большого порядка T можно получить если подавать на вход текущей модели выход предыдущей (см. рис. 1).

Policy gradient [20] алгоритм используется для обучения модели. В качестве состояния (state) выступает набор вероятностей трансформаций на каждом временном этапе. Чтобы получить награду (reward) от конкретной трансформации (action) считается разница показателей качества основной модели на исходном датасете с новым трансформированным

признаком и без него:

$$R_t = Q_t - Q_{t-1}.$$

Дисконтированная кумулятивной награда считается, как

$$R_t^{(k)} = R_t + \gamma R_{t+1} + \ldots + \gamma^k R_{t+k}.$$

Пусть

$$R_t^{\lambda} = (1-\lambda)\sum_{k=1}^{p\times T} \lambda^{k-1} R_t^{(k)},$$

тогда оптимизируется функционал качества

$$L(\theta) = -\mathbb{E}_{P\left(a_{1:p\times T};\theta\right)}\left[\sum_{t=1}^{p\times T} R_t^{\lambda}\right],$$

где θ — параметры модели. Далее применяется REINFORCE [21] правило и Монте-Карло оценка [22] для получения аппроксимации градиента. Т.к. количество моделей зависит линейно от количества признаков данный подход не применим к задачам с большим количеством признаков.

4.3. Scalable Automatic Feature Engineering Framework (SAFE)

Один из распространенных подходов получения новых признаков – взятие их из модели машинного обучения. Так, авторы SAFE с помощью бустинга находят полезные комбинации признаков и используют их для создания новых.

В частности, считается разница информационных энтропий (information gain ratio) узла-родителя и узлов-детей для каждого пути в дереве. Если разница большая, то такую комбинацию признаков сохраняем и делаем над ними трансформации. Для отбора признаков используется information value (IV) после бинаризации по порогу:

$$IV = \sum_i \left(\frac{n_p^i}{n_p} - \frac{n_n^i}{n_n} \right) \times \frac{n_p^i / n_p}{n_n^i / n_n},$$

где n_p, n_n — число положительных и отрицательных меток; n_p^i, n_n^i — число меток соответствующего класса, попавших в i-й бин. Если IV мало, такой признак отбрасывается. Слишком скоррелированные признаки также убираются.

4.4. AutoLearn

Предлагаемая авторами модель [23] в качестве отборщика признаков использует взаимную ифнормацию (Mutual Information) и корреляцию расстояния (distance correlation). Взаимная информация равна:

$$\mathrm{MI}(X;Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \mathbb{P}_{X,Y}(x,y) \log_2 \frac{\mathbb{P}_X(x)\mathbb{P}_Y(y)}{\mathbb{P}_{X,Y}(x,y)},$$

где $\mathbb{P}_{X,Y}(x,y)$ — вероятность встретить объект (x,y) в выборке, $\mathcal{Z}$ — множество значений переменной Z. Пусть

$$a_{i,j} = \|x_i - x_j\|_2, b_{i,j} = \|y_i - y_j\|_2, i,j = 1,2,\ldots,n,$$
$$A_{i,j} := a_{i,j} - \bar{a}_{i.} - \bar{a}_{.j} + \bar{a}_{..}, \quad B_{i,j} := b_{i,j} - \bar{b}_{i.} - \bar{b}_{.j} + \bar{b}_{..},$$

где $\bar{a}_{i.}$ — среднее значение i-ой строчки, $\bar{a}_{.j}$ — среднее значение j-го столбца.

$$\mathrm{Cov}_\mathrm{n}^2(X,Y) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n A_{i,j} B_{i,j}$$

Тогда квадрат корреляции расстояния равен

$$\mathrm{Cor}_\mathrm{n}^2(X,Y) = \begin{cases} \frac{\mathrm{Cov}_\mathrm{n}^2(X,Y)}{\sqrt{\mathrm{Cov}_\mathrm{n}^2(X,X)\,\mathrm{Cov}_\mathrm{n}^2(Y,Y)}}, & \mathrm{Cov}_\mathrm{n}^2(X,X)\,\mathrm{Cov}_\mathrm{n}^2(Y,Y) > 0 \\ 0, & \mathrm{Cov}_\mathrm{n}^2(X,X)\,\mathrm{Cov}_\mathrm{n}^2(Y,Y) = 0 \end{cases}$$

Алгоритм получения новых признаков состоит из следующих четырех основных этапов:

1. Предварительная обработка: оставляются только признаки с большой взаимной информацией (MI).
2. Определение скоррелированных признаков с помощью distance correlation. Данная оценка важности признака позволяет измерять любую нелинейную зависимость между признаками.

3. Построение новых признаков. Предскажем моделью один признак по-другому, а также добавим разницу между предсказаниями и истинной меткой, как признак. Если два исходных признака имеют большую distance correlation, то в качестве модели используется ядерная гребневая регрессия, иначе обычная гребневая регрессия.
4. Отбор признаков: делается с помощью случайного алгоритма лассо регрессии (RandomizedLasso) и использования взаимной информации по аналогии с п. 1.

4.5. LightAutoML (LAMA)

В данном методе берутся взаимодействия признаков, исходя из степени их кардинальности. После чего делается кодирование категориальных переменных целевым признаком (TE).

4.6. Sure Independence Screening and Sparsifying Operator (SISSO)

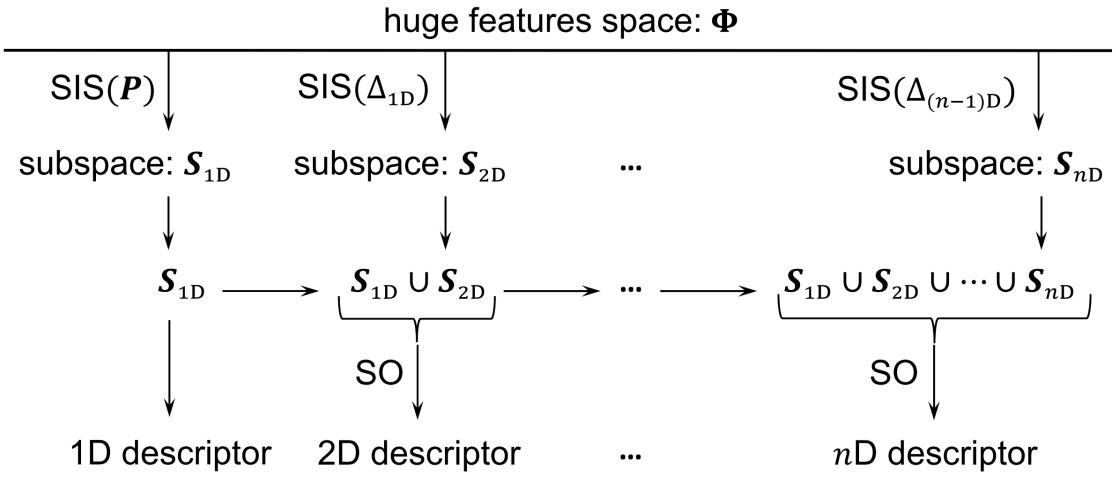

Рис. 2: Схема работы метода sisso [4]

Авторы данной статьи [4] используют итеративный подход для генерации новых признаков. Изначально генерируются всевозможные трансформации признаков заданного порядка (Ф см. рис. 2). После чего стоит задача отбора признаков в нужном количестве. Для промежуточной стадии отбора используется корреляция (SIS) с целевым признаком (P). Алгоритм нахождения новых признаков выглядит так:

1. Для первой итерации выделяется группа «хороших» признаков (S_{1p}) и из них лучший (1d descriptor).
2. Подсчитывается ошибка гребневой регрессии ($\Delta_{i\,\mathrm{D}}$) при предсказании целевого признака с помощью признаков, выступающих в качестве кандидатов.
3. Берутся топ признаков высоко скоррелированных с ошибкой на шаге 2. Данные признаки сохраняются (S_{ip}).
4. Из признаков $S_{(i-1)\,\mathrm{D}} \cup S_{ip}$ с помощью l_0 или l_1 регуляризации отбираются i признаков.

Таким образом алгоритм постепенно находит подпространства, наиболее связанные с целевым признаком.

4.7. Feature Engineering Wrapper (FEW)

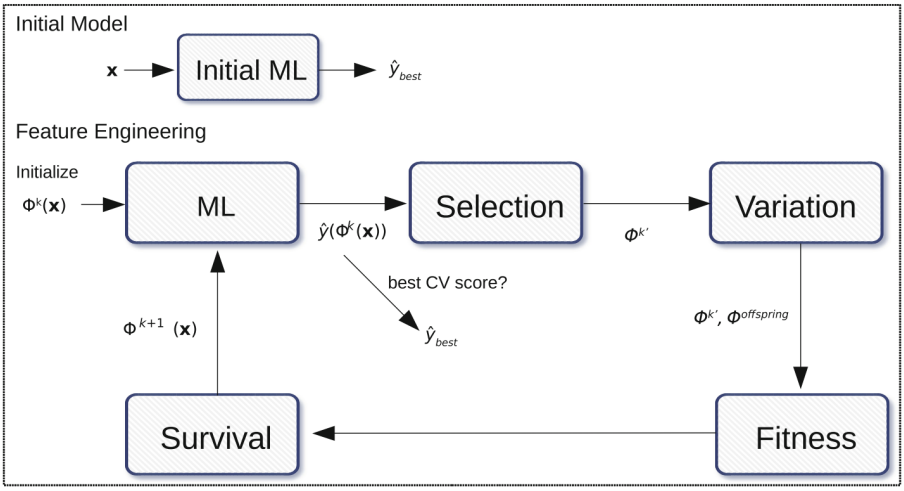

Рис. 3: Схема работы метода few [6]

Генетические алгоритмы применимы для получения новых признаков, если определить как происходит мутация и отбор поколений. Авторы данного метода использовали следующий подход (см. рис. 3):

1. Генерируются случайные группы трансформированных признаков (поколения).
2. Отбираются часть поколений по качеству решения задачи с помощью модели машинного обучения.
3. Текущие поколения подвергаются мутации:
   - кроссовер для дерева (tree crossover, см. рис. 4),
   - точечная мутация (point mutation, см. рис. 5).
4. Оценивается качество поколений.
5. Отбор поколений через эпсилон выживание (*eps*-lexicase survival).

Эпсилон выживание заключается в следующей процедуре:

1. Инициализация поколений G и множества S случайных выборок из данных.
2. Берутся топ поколений находящихся в *delta* окрестности по качеству на i-й выборки, принадлежащей S, где $delta = \text{median}(|metric - \text{median}(metric)|)$, $metric$ = массив всех показателей качества поколений на i-й выборке.
3. Переход на 2. Цикл прерывается по достижению нужного числа поколений или отсутствию новых выборок данных.

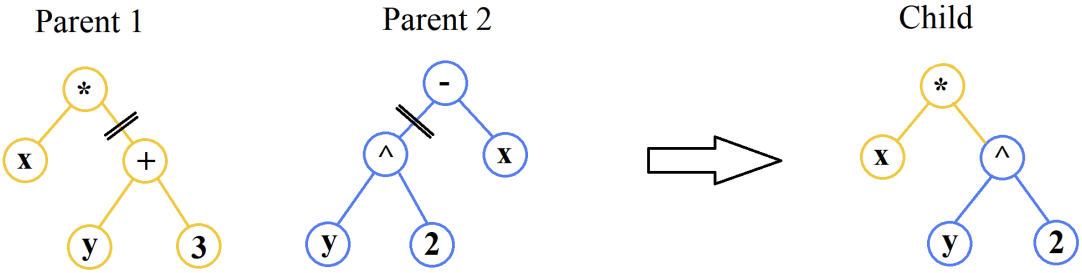

Рис. 4: Пример кроссовера для дерева.

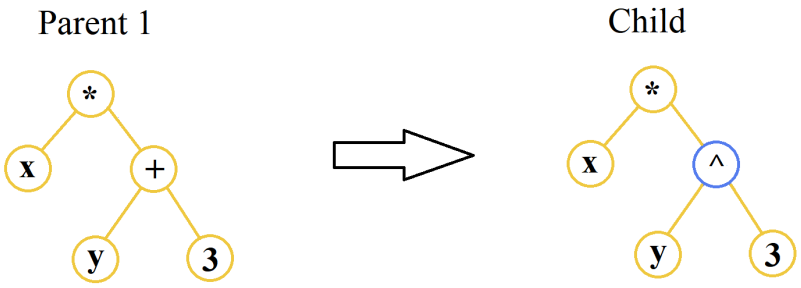

Рис. 5: Пример точечной мутации для дерева.

4.8. Iterative Feature Construction algorithm (TFC)

Данный алгоритм [24] заключается в итеративном построении новых трансформаций над уже существующими признаками и последующий их отбор с помощью теста хи-квадрат. В качестве трансформаций брались такие операции, как сложение, вычитание, умножение, деление. Исходные признаки могут быть полезны при комбинации с трансформированными, однако не гарантированно, что они пройдут многоэтапный отбор. Модифицированный метод выглядит следующим образом:

Algorithm 1 Модифицированный TFC

Вход:

$K \leftarrow$ количество итераций,
$(F, y) \leftarrow$ (признаки из датасета, целевая переменная)
$\mathrm{Tr} \leftarrow$ множество трансформаций,
$\chi^2 \leftarrow$ отбощик признаков на основе теста хи-квадрат,
$\mathrm{Tr}(F) \leftarrow$ применение трансформаций на всех парах признаков из F,
$\chi^2(F, y) \leftarrow$ отбор признаков из F.

Выход:

$F' \leftarrow$ новые признаки

Тело алгоритма:

$F' = F$

for $i = 1, ..., K$ **do**

$F' = F' \cup F$

$F' = \mathrm{Tr}(F')$

$F' = \chi^2(F', y)$

§5. Эксперименты

В качестве языка программирования для реализации методов автоматической генерации признаков и проведения экспериментов выбран python. Использовался docker для упрощения запуска экспериментов, так как не все методы исходно реализованы на python, например, safe написан на java. Часть методов, используемых или модифицированных нами, имеют определенные права, которые не позволяют их публиковать. Из-за этого принято решение выложить код всех методов на приватный github репозиторий [25].

5.1. Датасеты

Таблица 1: Датасеты для тестирования методов автоматической генерации признаков.

	name	**task**	**nrow**	**ncol**	**ncat**	**nrel**	**source**
1	mfeat-fourier	binary	2000	77	1	76	autolearn
2	spambase	binary	4601	58	1	57	safe
3	waveform-5000	binary	5000	41	1	40	autolearn
4	sylvine	binary	5124	21	1	20	openml
5	eeg-eye-state	binary	14980	15	1	14	safe
6	default-of-credit-card-clients	binary	30000	25	1	23	nfs
7	Amazon_employee_access	binary	32769	10	10	0	nfs
8	bank-marketing	binary	45211	17	10	7	openml
9	MiniBooNE	binary	130064	51	1	50	openml
10	airlines	binary	539383	8	5	3	openml

Часть датасетов была взята из открытого бенчмарка по автоматическому машинному обучению (OpenML AutoML) [26], другая часть из статьей. Если посмотреть на показатели датасетов (число строк, столбцов и тому подобное), то можно увидеть, что они сильно различаются, то есть данные репрезентативны. Для проведения эксперимента каждый датесет делится на обучающую (70%) и тестовую (30%) выборки.

5.2. Предобработка данных

В данных содержатся количественные и категориальные признаки. Необходимо каким-то образом закодировать категориальные переменные, так как чаще всего они содержатся в виде текста. Для бустинга использовалось количественное кодирование, а для линейной регрессии one-hot. Перед обучением метода автоматической генерации признаков и линейной регрессии количественные признаки стандартизовались. Перед работой всех моделей и методов пропуски в данных заполнялись медианным значением.

5.3. Алгоритмы

Реализация авторов таких методов, как autolearn, safe была некорректной. В autolearn не итерировался один из счетчиков цикла, что приводит к созданию не всех трансформаций признаков. Кроме того, при предсказании значений признака на тестовых данных не использовались данные из обучающей выборки. В safe обучение бустинга на валидационных данных делает модель переобученной.

Во всех экспериментах участвуют следующие модели:

- **бустинг (LightGBM)** с параметрами по умолчанию, за исключением параметров: bagging_freq=1, metric=auc, num_boost_round=10000, early_stopping_rounds=10,
- **линейная модель (Logistic Regression)** с параметрами по умолчанию, за исключением параметра max_iter=1000.

Параметры выбраны так, чтобы модели не получились слишком простые. Обучение моделей происходило по кросс-валидации с пятью фолдами. Параметры оптимизировались с помощью optuna [27] на одном фолде с ограничениями в 100 попыток и 30 минут для максимизации показателя качества auc по следующей сетке гиперпараметров:

Таблица 2: Оптимизируемые гиперпараметры бустинга, нижняя и верхняя границы.

name	**low value**	**high value**
num_leaves	2	256
feature_fraction	0.4	1.0
bagging_fraction	0.4	1.0

Таблица 3: Оптимизируемый гиперпараметр логистической регрессии по логарифмической шкале, нижняя и верхняя границы.

name	**low value**	**high value**
C	1e-5	20.0

Подбор гиперпараметров (см. таблицы 2, 3) делает модели менее переобученными.

Критерий качества auc соответствует общепринятому показателю качества для задачи бинарной классификации на открытом бенчмарке по автоматическому машинному обучению [26]. В статьях [12], [28], [23] по автоматической генерации признаков еще используют сбалансированную точность, f1.

В качестве методов автоматической генерации признаков выступают:

- **autofeat**,
- **nfs**,
- **safe**,
- **autolearn**,
- **sisso**,
- **few**,
- **lama**,
- **tfc**.

Во всех методах использовались параметры из соответствующих статей.

5.4. Результаты

5.4.1 Lgbm

data_ind	base	lama	safe
1	1.0	**1.0**	0.999
2	**0.989**	0.988	0.985
3	0.955	**0.957**	0.957
4	0.983	0.983	**0.992**
5	**0.989**	0.989	0.988
6	0.772	**0.776**	0.767
7	0.858	**0.871**	0.818
8	**0.932**	0.925	0.926
9	0.985	**0.986**	0.985
10	**0.722**	0.722	0.716

Таблица 4: Показатель качества auc для модели lgbm в завивимости от датасета и метода автоматической генерации признаков.

data_ind	base	lama	safe
1	**0.983**	0.889	0.983
2	**0.949**	0.944	0.945
3	0.825	**0.832**	0.825
4	0.943	0.942	**0.955**
5	**0.945**	0.943	0.934
6	0.531	**0.536**	0.528
7	**0.975**	0.974	0.973
8	**0.626**	0.603	0.604
9	0.905	**0.906**	0.904
10	**0.655**	0.654	0.651

Таблица 5: Показатель качества f1 для модели lgbm в завивимости от датасета и метода автоматической генерации признаков.

SAFE явно дает меньше информации в отличии от LAMA. В большинстве случаев бустингу среднее кодирование дает существенно больший прирост в качестве нежели комбинации признаков, полученные после бинарных трансформаций. Однако среднее кодирование может повысить шанс переобучиться, что не всегда дает уверенность в лучшем результате.

5.4.2 Linear

data_ind	base	lama	safe
1	**1.0**	**1.0**	**1.0**
2	0.965	**0.967**	0.967
3	0.927	0.927	**0.949**
4	0.962	0.962	**0.973**
5	0.675	**0.755**	0.602
6	0.714	0.714	**0.745**
7	0.537	**0.858**	0.605
8	**0.909**	0.902	0.907
9	0.955	0.956	**0.963**
10	0.688	**0.714**	0.684

Таблица 6: Показатель качества auc для модели linear в зависимости от датасета и метода автоматической генерации признаков.

data_ind	base	lama	safe
1	**1.0**	**1.0**	0.974
2	0.898	**0.912**	0.906
3	–	–	**0.832**
4	0.918	0.916	**0.928**
5	0.599	**0.688**	0.581
6	0.51	0.508	**0.51**
7	0.97	**0.972**	0.97
8	**0.58**	0.565	0.565
9	0.826	0.831	**0.85**
10	0.639	**0.651**	0.638

Таблица 7: Показатель качества f1 для модели linear в зависимости от датасета и метода автоматической генерации признаков.

Линейная регрессия явно лучше работает на трансформированных признаках. Если смотреть на показатель качества auc, то бинарные трансформации признаков лучше работают, чем среднее кодирование. Учитывая, что категориальные признаки подаются в one-hot кодировании, полезность новых признаков становится очевидной.

§6. Заключение

На защиту выносится:

1. Реализация таких методов автоматической генерации признаков, как autolearn, tfc.
2. Доработка nfs, safe.
3. Добавлен интерфейс на языке python ко всем использованным методам, а также есть возможность запуска эксперимента в docker окружении.
4. Проведены эксперименты на искусственных и реальных данных.

§7. Список литературы

1. *He X.*, *Zhao K.*, *Chum X.* AutoML: A survey of the state-of-the-art // Knowledge-Based Systems. — 2021. — Vol. 212. — P. 106622.
2. *Katz G.*, *Shin E.*, *Song D.* Explorekit: Automatic feature generation and selection // IEEE 16th International Conference on Data Mining (ICDM). — 2016. — P. 979–984.
3. *Horn F.*, *Pack R.*, *Rieger M.* The autofeat python library for automated feature engineering and selection // Joint European Conference on Machine Learning and Knowledge Discovery in Databases. — 2019. — P. 111–120.
4. *Ouyang R.* SISSO: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates // Physical Review Materials. — 2018. — Vol. 2, no. 8. — P. 083802.
5. *Luo Y.* Autocross: Automatic feature crossing for tabular data in real-world applications // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining. — 2019. — P. 1936–1945.
6. *La C.*, *Moore J.* General Feature Engineering Wrapper for Machine Learning Using ϵ-Lexicase Survival // European Conference on Genetic Programming. — 2017. — P. 80–95.
7. *Olson R.*, *Moore J.* TPOT: A tree-based pipeline optimization tool for automating machine learning // Workshop on automatic machine learning. – PMLR. — 2016. — P. 66–74.
8. *Shi Q.* Scalable automatic feature engineering framework for industrial tasks // IEEE 36th International Conference on Data Engineering (ICDE). — 2020. — P. 1645–1656.
9. *Zhang J.*, *Hao J.* Cross-data Automatic Feature Engineering via Meta-learning and Reinforcement Learning // Pacific-Asia Conference on Knowledge Discovery and Data Mining. – Springer. — 2020. — P. 818–829.
10. *Zhang J.* Automatic feature engineering by deep reinforcement learning // Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. — 2019. — P. 2312–2314.
11. *Khurana U.* Cognito: Automated feature engineering for supervised learning // IEEE 16th International Conference on Data Mining Workshops (ICDMW). – IEEE. — 2016. — P. 1304–1307.
12. *Chen X.* Neural feature search: A neural architecture for automated feature engineering // IEEE International Conference on Data Mining (ICDM). – IEEE. — 2019. — P. 71–80.
13. *Xie Y.* Fives: Feature interaction via edge search for large-scale tabular data // Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining. — 2021. — P. 3795–3805.
14. *Liu Z.* Dnn2lr: Interpretation-inspired feature crossing for real-world tabular data // arXiv preprint arXiv:2008.09775. — 2020.
15. *Abdi H.*, *Williams L.* Principal component analysis // Wiley interdisciplinary reviews: computational statistics. — 2010. — Vol. 2, no. 4. — P. 433–459.
16. *Torkkola K.* Feature extraction by non-parametric mutual information maximization // Journal of machine learning research. — 2003. — Vol. 3. — P. 1415–1438.
17. *Duan J.* Automated generation and selection of interpretable features for enterprise security // IEEE International Conference on Big Data (Big Data). – IEEE. — 2018. — P. 1258–1265.

18. *Székely G.*, *Rizzo M.*, *Bakirov N.* Measuring and testing dependence by correlation of distances // The annals of statistics. — 2007. — Vol. 35, no. 6. — P. 2769–2794.

19. *Neshatian K.*, *Zhang M.*, *Johnston M.* Feature construction and dimension reduction using genetic programming // Australasian Joint Conference on Artificial Intelligence. – Springer. — 2007. — P. 160–170.

20. Policy gra- dient methods for reinforcement learning with function approximation / R. Sutton [et al.] // in Proceedings of NIPS. — 1999. — P. 1057–1063.

21. *Williams R. J.* Simple statistical gradient-following algorithms for connectionist reinforcement learning // Machine learning. — 1992. — Vol. 8, no. 3. — P. 229–256.

22. *Robert P. C.*, *Casella G.* Monte Carlo Statistical Methods // Springer Texts in Statistics. — 2004.

23. *Kaul A.*, *Maheshwary S.*, *Pudi V.* Autolearn—Automated feature generation and selection // IEEE International Conference on data mining (ICDM). — 2017. — P. 217–226.

24. *Piramuthu S.*, *Sikora R.* Iterative feature construction for improving inductive learning algorithms // Expert Systems with Applications. — 2009. — Vol. 36, no. 2. — P. 3401–3406.

25. *Kuznetsov M.* Accompanying repository: Automatic feature generation for tabular data // URL: https://github.com/MikhailKuz/afg. — 2022.

26. *Gijsbers P.* An open source AutoML benchmark // arXiv preprint arXiv:1907.00909. — 2019.

27. *Akiba T.* Optuna: A next-generation hyperparameter optimization framework // Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery data mining. — 2019. — P. 2623–2631.

28. *Dor O.*, *Reich Y.* Strengthening learning algorithms by feature discovery // Information Sciences. — 2012. — Vol. 189. — P. 176–190.