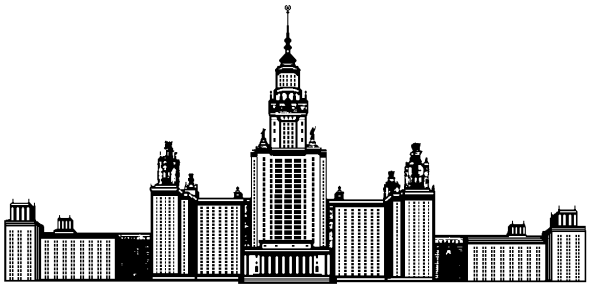

Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

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. 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 чистка, генерирование новых признаков, 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.

В области автоматического машинного обучения (automl) умная генерация признаков — один из ключевых компонентов.

В данной работе проводятся эксперименты над методами автоматической генерации признаков на табличных данных.

§2. Обзор литературы

Основное деление методов:

- Генерация-отбор или итеративный (expand reducion):
  - все сразу генерируем признаки [4], [3], [12],
  - только перспективные [9].
- Генетический [7], [11].
- С помощью дерева [15], [18], [17], [6].
- С помощью нейросетей [1], [16], [8].

Деление трансформаций:

- Унарные:
  - применение нелинейных функций: exp(x), log(1 + x).
- Бинарные:
  - +, -, /, *,
  - агрегация (groupby + func).
- Многоуровневые:
  - кодирование категориальных признаков (целевое кодирование (TE), количественное кодирование (count encoding)),
  - методы снижения размерности (AE, PCA, t-SNE, MMI),
  - knn признаки в задаче без учителя (расстояния между наблюдениями, признаки ближайших наблюдений),
  - предсказание предыдущих признаков,
  - 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 [2].

Большинство методов, основанных на генерации-отборе, имеют встроенный отборщик признаков. В качестве такого выступают:

- Взаимная информация.
- Корреляция и корреляция расстояния (distance correlation).
- Веса линейной модели.
- Оценка важности мета-моделью.

Например, в ExploreKit [4] 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[7] в отличии от ядрового метода имеет сложность, зависящую линейно от количества объектов в выборке. Это позволяет применять его на больших данных.

FCDRGP [10] полезен для бустинга, так как он подбирает признаки исходя из 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.

Преимущество SAFE [4] 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 $(x, y) = (x_1, x_2, ..., x_D, y)$. 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 $\phi : \mathcal{S} \to \mathbb{R}^d$, которая возвращает трансформированные данные.

Естественно, интерес заключается в нахождении новых признаков, которые:

- улучшают показатель качества,
- имеет небольшую размерность,
- быстро считаются,
- полезны для многих моделей машинного обучения.

§4. Методы решения

4.1. Autofeat

Авторы данной статьи [3] сначала применяют классический подход: генерация всевозможные 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.
2. Корреляция с другими признаками должна быть меньше определенного порога.

3. Для поиска на немного менее важных признаков брались выборки 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 отсеиваются ненужные признаки.

4.2. Neural Feature Search (NFS)

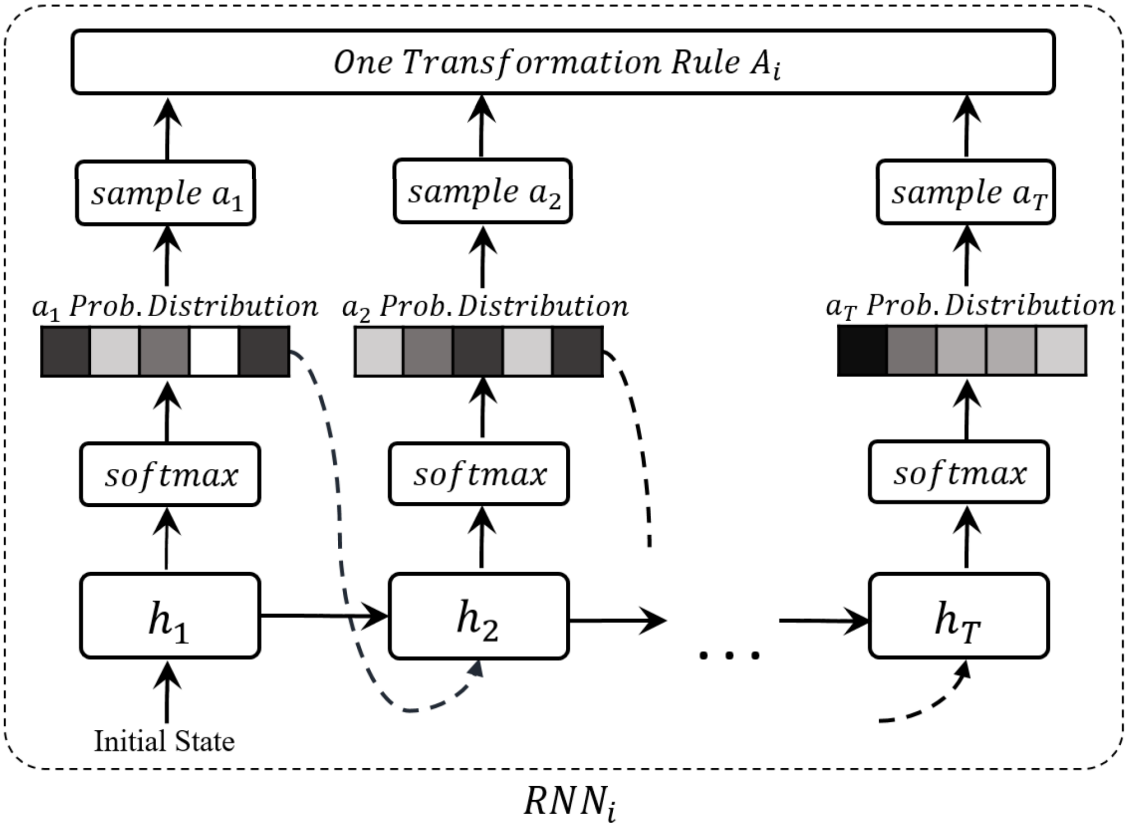

Рис. 1: Схема работы нейронной сети для генерации трансформаций в методе nfs [1]

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

Policy gradient [14] алгоритм используется для обучения модели. Чтобы получить награду от конкретной трансформации считается показатель качества основной модели на исходном датасете и датасете с новым трансформированным признаком.

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:

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

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 [5] состоит из следующих четырех основных этапов:

1. Предварительная обработка: оставляются только признаки с большой взаимной информацией.
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)

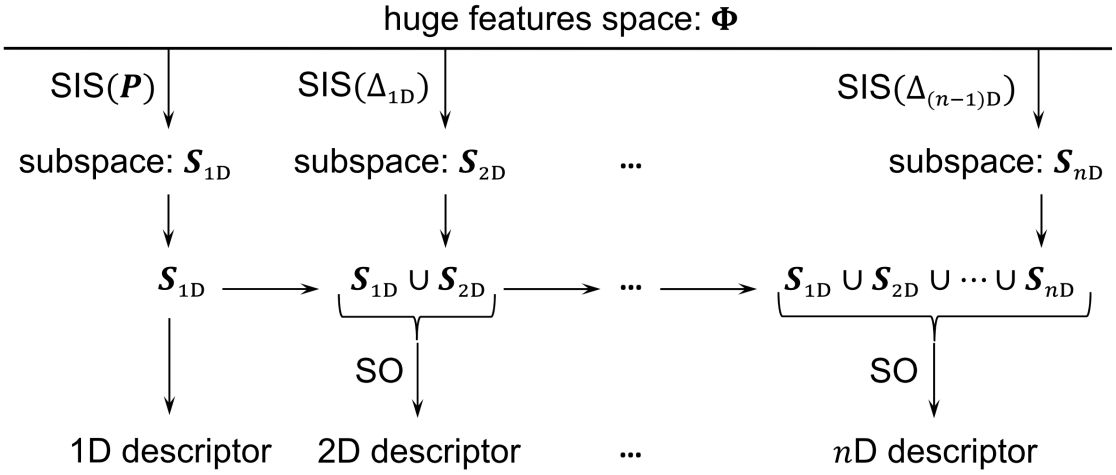

Рис. 2: Схема работы метода sisso [12]

Авторы данной статьи [12] 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_{1D}) и из них лучший (1d descriptor).
2. Подсчитывается ошибка гребневой регрессии (Δ_{id}) 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_{iD}).
4. Из признаков $S_{(i-1)D} \cup S_{iD}$ 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)

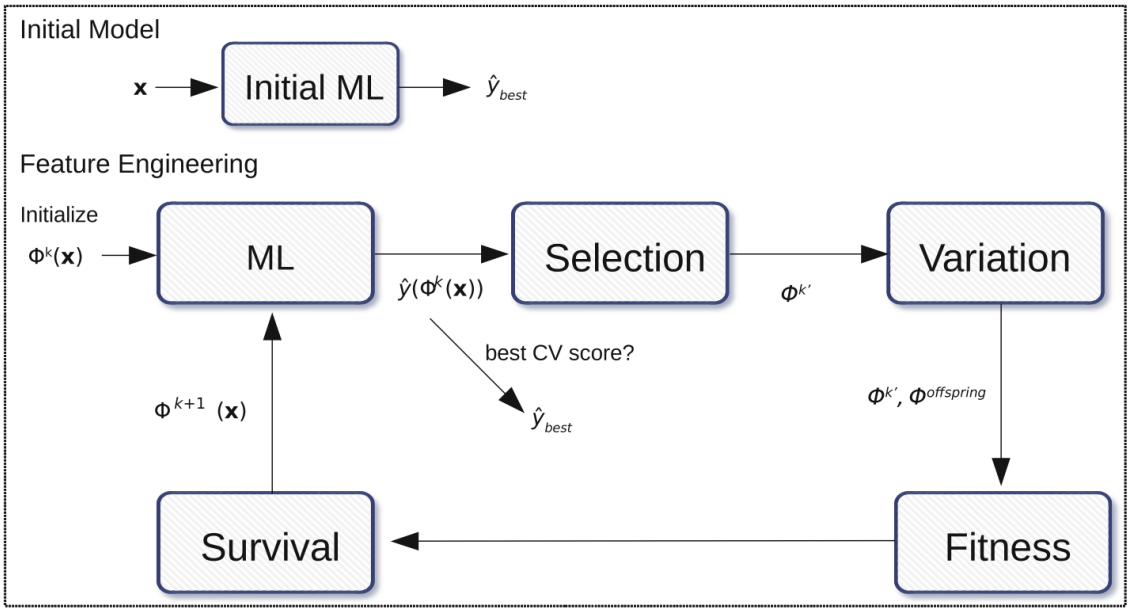

Рис. 3: Схема работы метода few [7]

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).
   - Точечная мутация (point mutation).
4. Оценивается качество поколений.
5. Отбор поколений через эпсилон выживание (*eps*-lexicase survival).

Эпсилон выживание представляет из себя следующую процедуру:

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.

4.8. Iterative Feature Construction algorithm (TFC)

Данный алгоритм [13] 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.

§5. Эксперименты

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), другая часть из статьей с неправильной реализацией экспериментов. Если посмотреть на показатели датасетов (число строк, столбцов и тому подобное), то можно увидеть, что они сильно различаются.

5.2. Предобработка данных

В данных содержаться 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. При обучении линейной регрессии данные стандартизовались.

5.3. Алгоритмы

Во всех экспериментах участвуют следующие модели:

- **бустинг (LightGBM)**,
- **линейная модель (Lr)**.

Параметры оптимизировались с помощью optuna 30 минут для максимизации показателя качества auc.

В качестве методов автоматической генерации признаков выступают:

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

- **tfc**.

Во всех методах использовались параметры из соответствующих статей. В большинстве случаев использовалось масштабирование. Для линейной модели имеет смысл преобразовывать категориальные переменные с помощью one-hot кодирования, для бустинга — количественным кодированием.

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 2: 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 3: Показатель качества f1 для модели lgbm в завивимости от датасета и метода автоматической генерации признаков.

SAFE явно не дает меньше информации в отличии от LAMA.

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 4: 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 5: Показатель качества f1 для модели linear в зависимости от датасета и метода автоматической генерации признаков.

Линейная регрессия работает лучше как на признаках LAMA, так и на признаках SAFE.

§6. Заключение

В данной работе рассмотрены различные методы автоматической генерации признаков на табличных данных. Реализованы такие методы, как autolearn, tfc. Доработаны nfs, safe. Добавлен интерфейс на языке python ко всем использованным методам, а также есть возможность запуска эксперимента в docker окружении. Проведены эксперименты на искусственных и реальных данных. Можно сделать следующие выводы:

- 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.
- 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.

§7. Список литературы

[1] Chen X. Neural feature search: A neural architecture for automated feature engineering // IEEE International Conference on Data Mining (ICDM). – IEEE. — 2019. — P. 71–80.

[2] 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.

[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] Katz G., Shin E., Song D. Explorekit: Automatic feature generation and selection // IEEE 16th International Conference on Data Mining (ICDM). — 2016. — P. 979–984.

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

[6] Khurana U. Cognito: Automated feature engineering for supervised learning // IEEE 16th International Conference on Data Mining Workshops (ICDMW). – IEEE. — 2016. — P. 1304–1307.

[7] La Cava., Moore J. General Feature Engineering Wrapper for Machine Learning Using ϵ-Lexicase Survival // European Conference on Genetic Programming. — 2017. — P. 80–95.

[8] Liu Z. Dnn2lr: Interpretation-inspired feature crossing for real-world tabular data // arXiv preprint arXiv:2008.09775. — 2020.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] Policy gra- dient methods for reinforcement learning with function approximation / Sutton R., McAllester D., Singh A., and Mansour Y. // in Proceedings of NIPS. — 1999. — P. 1057–1063.

[15] Shi Q. Scalable automatic feature engineering framework for industrial tasks // IEEE 36th International Conference on Data Engineering (ICDE). — 2020. — P. 1645–1656.

[16] 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.

[17] 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.

[18] 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.