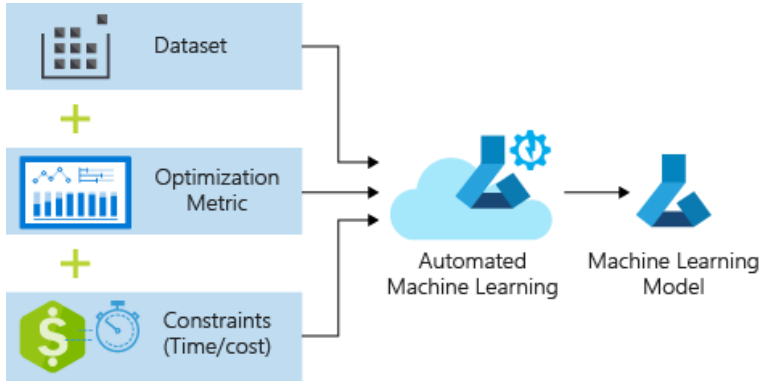
AutoML

Елена Кантонистова

Что такое AutoML?

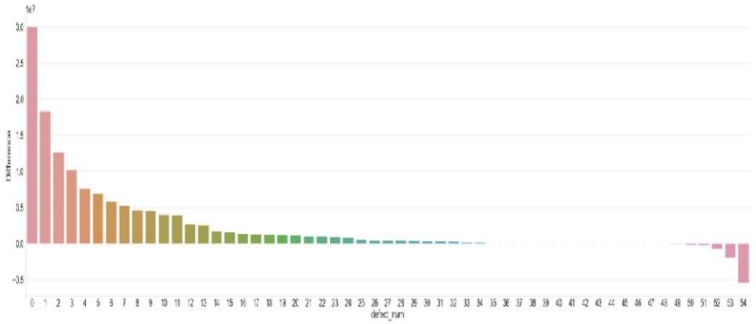
- В классическом машинном обучении Data Scientist делает все шаги пайплайна самостоятельно, а именно:
- подготавливает данные: заполняет пропуски, кодирует категориальные признаки
- занимается feature engineeringom: придумывает новые информативные признаки
- снижает размерность: отбирает признаки или применяет методы для снижения размерности
- обучает модели: обучает модели и подбирает их гиперпараметры
- выбирает итоговый пайплайн (обработка данных + модель) или же ансамбль пайплайнов/моделей.



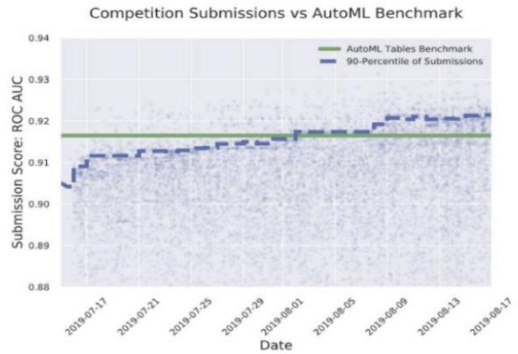
Эта работа достаточно трудоемкая и ресурсозатратная. При этом все эти шаги ожидаются от специалиста в каждой задаче, поэтому были разработаны библиотеки для автоматического машинного обучения (AutoML), автоматизирующие один или несколько этих шагов.

Зачем нужен AutoML?

- Зачем нужен AutoML, если есть Data Scientist? Попробуем ответить на этот вопрос:
- Во-первых, мы экономим ценный ресурс - время специалиста и деньги на его оплату. Специалист теперь может заниматься более творческими задачами, а рутину оставлять машине.
- Во-вторых, современные Фреймворки для AutoML показывают очень и очень неплохое качество работы! Обойти их по качеству могут только топовые специалисты (топ-10% среди всех data scientistов). Поэтому здесь мы экономим еще и деньги на оплату специалистов.



ML теперь выгоден там, где ранее не окупался



Автоматическое решение обходят только ТОП специалисты или для этого требуется время

Классификация AutoML

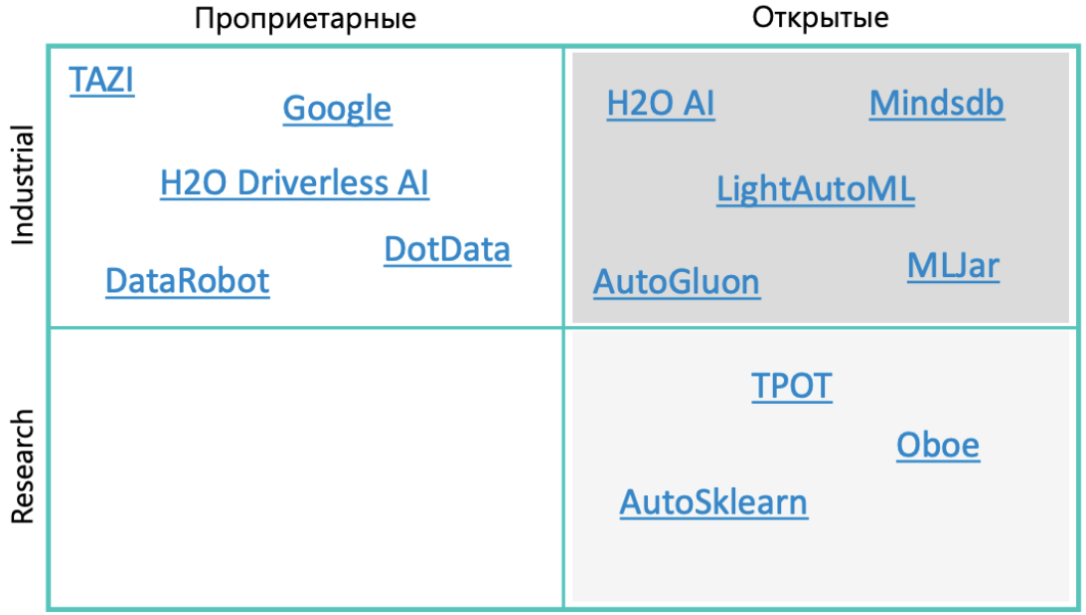
Шесть уровней AutoML:

1. Отсутствие автоматизации. Даже модели вы пишете собственноручно, с нуля.
2. Использование высокоуровневых API: sklearn, pandas, H2O, XGBoost и так далее.
3. Автоматический подбор гиперпараметров и ансамблирование. Базовый отбор моделей.
4. Автоматическая работа с данными: feature engineering, feature selection, data augmentation.
5. Автоматическая работа с признаками, зависящая от области и задачи.
6. Полная автоматизация решения (super human).

На данном этапе развития автоматического машинного обучения мы находимся между пунктами 4 и 5.

Обзор AutoML фреймворков

- Академические решения вроде TPOT, Oboe и FEDOT отлично подходят для экспериментов. Они хорошо показывают себя на задачах, где данных не очень много или время не слишком критично.
- Индустриальные фреймворки вроде H2O и AutoGluon и LAMA заточены под быстрое решение прикладных задач, в том числе на датасетах сравнительно большого объёма. Как правило, они имеют фиксированный пайплайн и больше полагаются на эвристические подходы. Такие фрейморки автоматизируют разработку моделей в широком смысле: отчёты, интеграционные обвязки и так далее. Разумеется, передовые подходы из академических решений со временем находят применение в индустриальной среде, помогая решать бизнес-задачи.

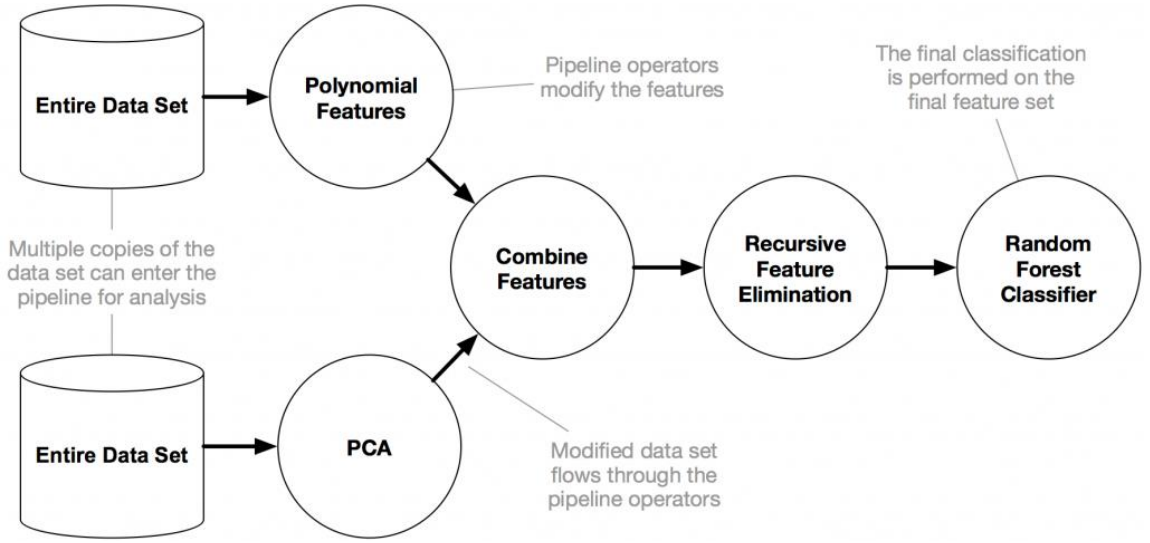


TPOT(Tree-based Pipeline Optimization Tool,2016, 2020)

TPOT - это фреймворк для автоматического выбора моделей машинного обучения и их гиперпараметров. Он использует *генетическое программирование* для эффективного поиска оптимальных конфигураций моделей. Фреймворк появился в 2016 году и был доработан в 2020 году.

Далее ***пайплайном*** будем называть некоторую последовательность преобразований исходного набора данных, включающую работу с признаками (генерацию, отбор, очистку и тд), а также выбор и обучение ML-модели.

В TPOT реализованы tree-based пайплайны, то есть схемы работы с данными и применения моделей, которые можно представить в виде деревьев.

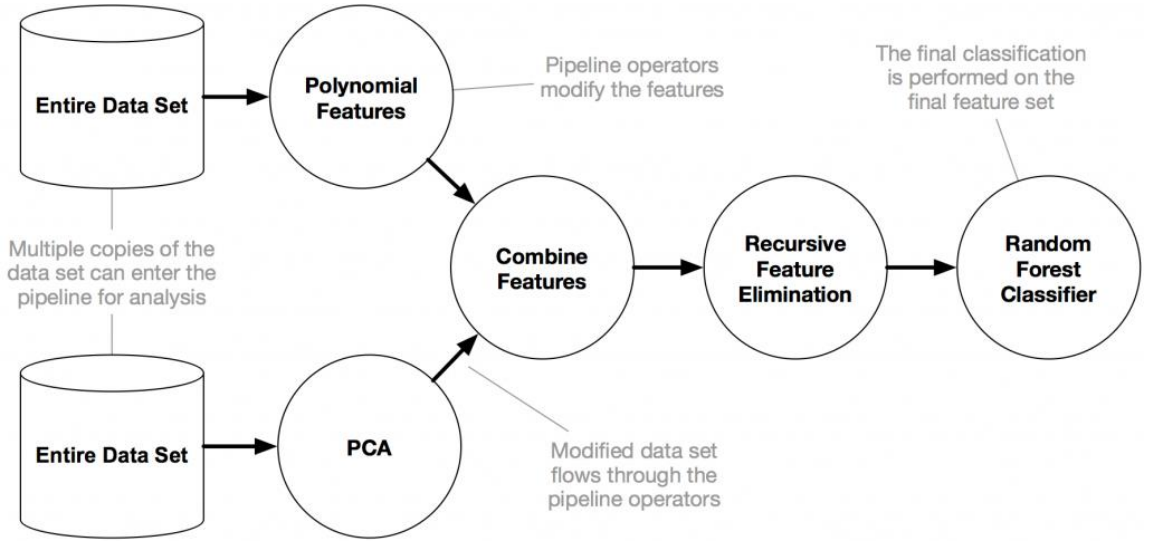


TPOT(Tree-based Pipeline Optimization Tool,2016, 2020)

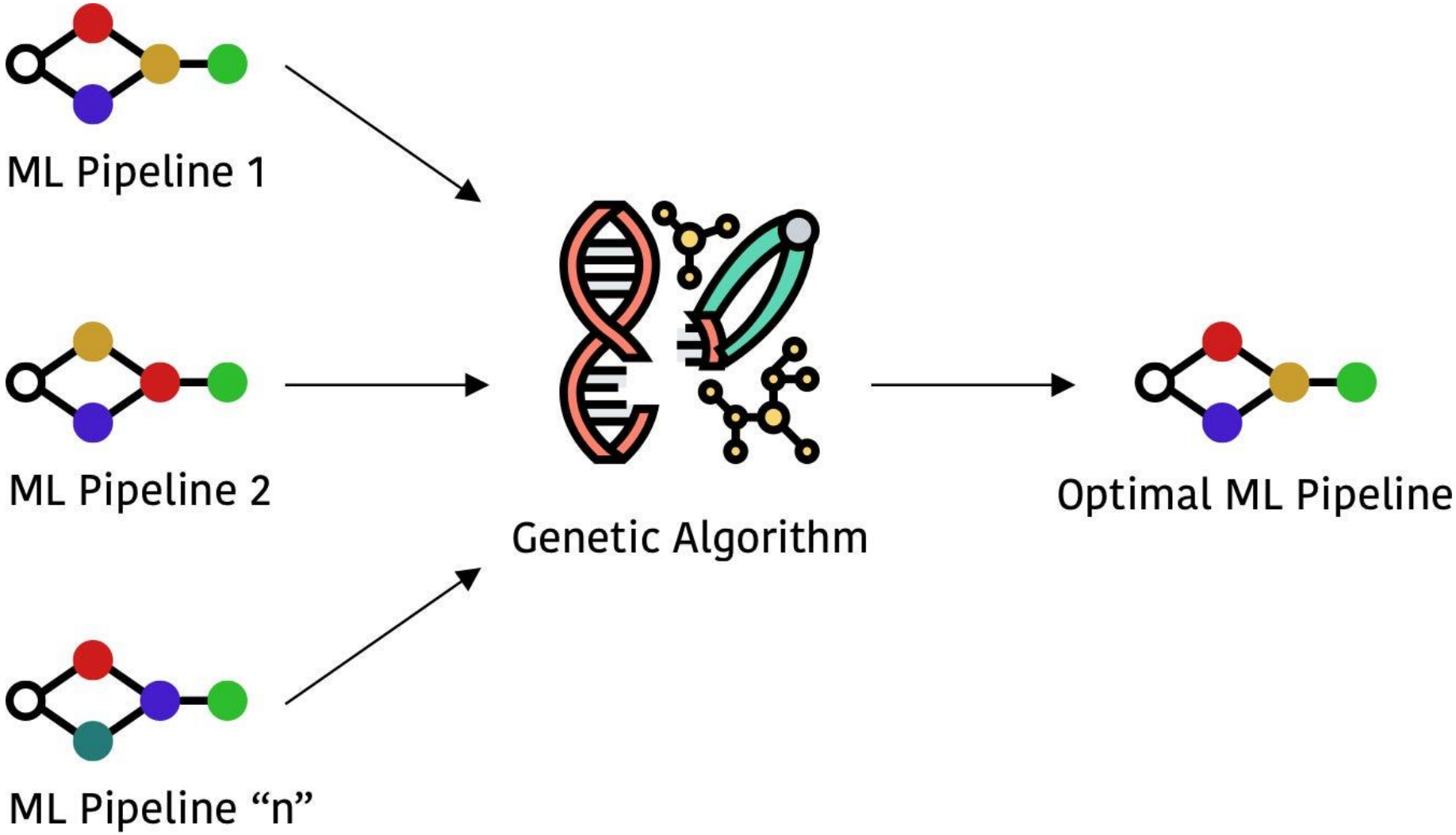
TPOT - это фреймворк для автоматического выбора моделей машинного обучения и их гиперпараметров. Он использует *генетическое программирование* для эффективного поиска оптимальных конфигураций моделей. Фреймворк появился в 2016 году и был доработан в 2020 году.

Далее ***пайплайном*** будем называть некоторую последовательность преобразований исходного набора данных, включающую работу с признаками (генерацию, отбор, очистку и тд), а также выбор и обучение ML-модели.

В TPOT реализованы tree-based пайплайны, то есть схемы работы с данными и применения моделей, которые можно представить в виде деревьев.



Алгоритм работы TPOT



Алгоритм работы TPOT

1. TPOT начинает с **генерации случайных пайплайнов**, которые представляют собой последовательность шагов обработки данных и моделей машинного обучения. Каждый пайплайн состоит из преобразований данных (например, масштабирование, преобразование признаков) и модели машинного обучения.

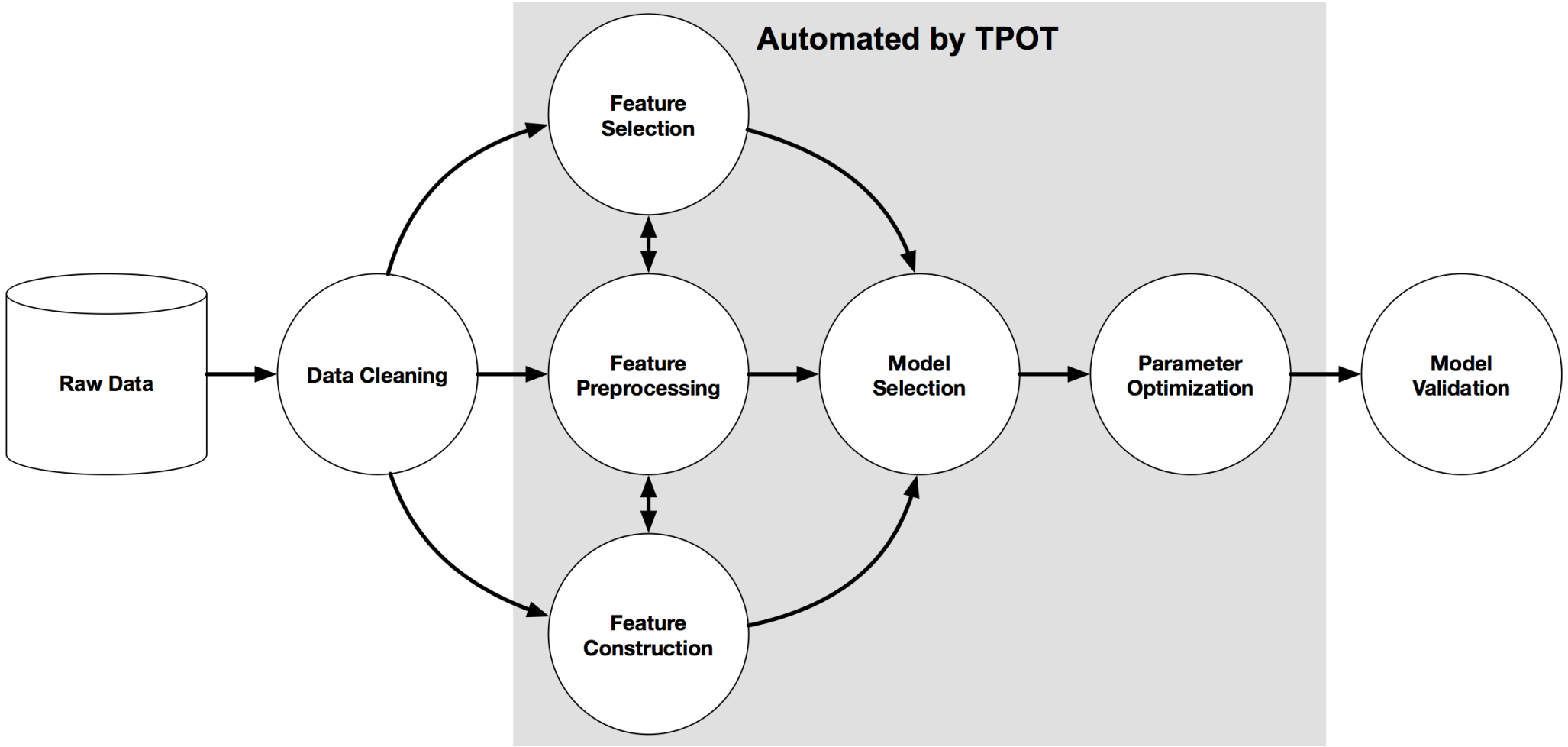
2. Каждый сгенерированный пайплайн **оценивается на наборе данных с использованием заданной метрики качества** (например, точность, F1-мера и тд). Эти оценки используются для ранжирования конвейеров.

3. TPOT применяет **генетическое программирование** для эффективного поиска оптимальных конфигураций. Он создает новые поколения пайплайнов, комбинируя и изменяя лучшие конвейеры из предыдущих поколений. Этот процесс включает в себя операции *скрещивания (комбинирование частей пайплайнов)* и *мутации (внесение случайных изменений)*.

4. На каждом шаге (поколении пайплайнов) происходит **отбор лучших пайплайнов**, основанный на их качестве. Лучшие пайплайны передаются в следующее поколение для создания новых вариантов.

5. Алгоритм **останавливается после заданного числа итераций или когда происходит сходимость к оптимальной конфигурации, либо же при достижении некоторого заранее заданного значения метрики**.

Какая часть ML-пайплайна автоматизируется TPOT?



TPOT: практика

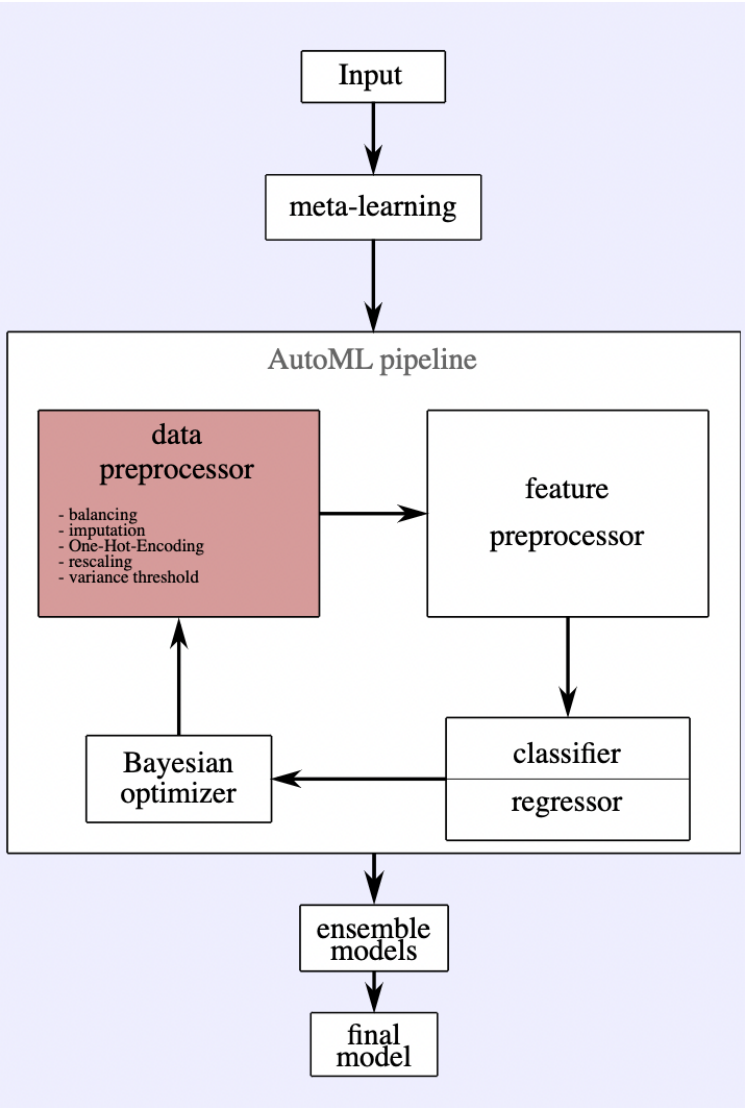
https://colab.research.google.com/drive/16wRV5sEOx0LMlZIc1PmQw1D_2stZPUi6

AutoSklearn (2015)

AutoSklearn - это библиотека для автоматического машинного обучения, в которой реализован автоматический выбор лучшего алгоритма из представленных в scikit-learn, а также настройка его гиперпараметров. Для улучшения обобщающей способности используются ансамбли из моделей, которые были получены в ходе оптимизации. В Auto-sklearn применяются идеи мета-обучения, которые позволяют выделять похожие датасеты и использовать знания о них.

Опишем общую схему работы AutoSklearn.

- Сначала фреймворк использует мета-обучение для выбора пайплайнов-кандидатов (шаги предобработки данных, модели и их гиперпараметры).
- Фреймворк AutoSklearn итеративно тестирует найденные пайплайны (обработка данных +предобработка признаков + обучение классификатора/регрессора): результаты оцениваются, а затем гиперпараметры оптимизируются с использованием байесовского оптимизатора.
- Лучшие модели ансамблируются (если это необходимо).



Metalearning в AutoSklearn

На первом этапе в AutoSklearn используется Meta-learning - это знания о моделях, которые хорошо работают на других датасетах.

В чем заключается алгоритм Meta-learning:

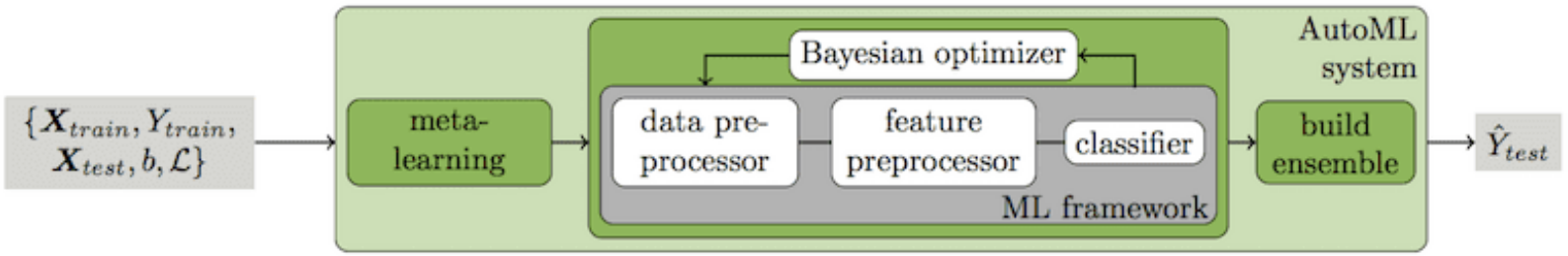
- Разработчики AutoSklearn обучили и настроили пайплайны (обработку данных + модели) на 140 открытых датасетах для задачи классификации из репозитория AutoML
- Для каждого из этих датасетов были вычислены мета-признаки (38 признаков), характеризующие датасет - число объектов, число признаков, числовые характеристики признаков и целевой переменной и так далее (подробнее смотри скриншот ниже)
- Затем из 140 стандартных датасетов выбраны датасеты, наиболее похожие на наш (на тот, к которому применяется AutoSklearn) по L1-мере, посчитанной на мета-признаках
- Отобраны пайплайны, лучше всего работающие на выбранных датасетах
- С этими пайплайнами далее работает AutoSklearn

Meta-feature	Value			Calculation time (s)		
	Minimum	**Mean**	**Maximum**	**Minimum**	**Mean**	**Maximum**
class-entropy	0.64	1.92	4.70	0.00	0.00	0.00
class-probability-max	0.04	0.43	0.90	0.00	0.00	0.00
class-probability-mean	0.04	0.28	0.50	0.00	0.00	0.00
class-probability-min	0.00	0.19	0.48	0.00	0.00	0.00
class-probability-std	0.00	0.10	0.35	0.00	0.00	0.00
dataset-ratio	0.00	0.06	0.62	0.00	0.00	0.00
inverse-dataset-ratio	1.62	141.90	1620.00	0.00	0.00	0.00
kurtosis-max	-1.30	193.43	4812.49	0.00	0.01	0.05
kurtosis-mean	-1.30	24.32	652.23	0.00	0.01	0.05
kurtosis-min	-3.00	-0.59	5.25	0.00	0.01	0.05
kurtosis-std	0.00	48.83	1402.86	0.00	0.01	0.05
landmark-1NN*	0.20	0.79	1.00	0.01	0.61	8.97
landmark-decision-node-learner*	0.07	0.55	0.96	0.00	0.13	1.34
landmark-decision-tree*	0.20	0.78	1.00	0.00	0.49	5.23
landmark-lda*	0.26	0.79	1.00	0.00	1.39	70.08
landmark-naive-bayes*	0.10	0.68	0.97	0.00	0.06	1.05
landmark-random-node-learner*	0.07	0.47	0.91	0.00	0.02	0.26
log-dataset-ratio	-7.39	-3.80	-0.48	0.00	0.00	0.00
log-inverse-dataset-ratio	0.48	3.80	7.39	0.00	0.00	0.00
log-number-of-features	1.10	2.92	5.63	0.00	0.00	0.00
log-number-of-instances	4.04	6.72	9.90	0.00	0.00	0.00
number-of-Instances-with-missing-values	0.00	96.00	2480.00	0.00	0.00	0.01
number-of-categorical-features	0.00	13.75	240.00	0.00	0.00	0.00
number-of-classes	2.00	6.58	28.00	0.00	0.00	0.00
number-of-features	3.00	33.91	279.00	0.00	0.00	0.00
number-of-features-with-missing-values	0.00	3.54	34.00	0.00	0.00	0.00
number-of-instances	57.00	2126.33	20000.00	0.00	0.00	0.00
number-of-missing-values	0.00	549.49	22175.00	0.00	0.00	0.00
number-of-numeric-features	0.00	20.67	216.00	0.00	0.00	0.00
pca-95percent*	0.02	0.52	1.00	0.00	0.00	0.00
pca-kurtosis-first-pc*	-2.00	13.38	730.92	0.00	0.00	0.01
pca-skewness-first-pc*	-27.07	-0.16	6.46	0.00	0.00	0.04
percentage-of-Instances-with-missing-values	0.00	0.14	1.00	0.00	0.00	0.00
percentage-of-features-with-missing-values	0.00	0.16	1.00	0.00	0.00	0.00
percentage-of-missing-values	0.00	0.03	0.65	0.00	0.00	0.00
ratio-categorical-to-numerical	0.00	1.35	33.00	0.00	0.00	0.00
ratio-numerical-to-categorical	0.00	0.49	7.00	0.00	0.00	0.00
skewness-max	0.00	5.34	67.41	0.00	0.00	0.04
skewness-mean	-0.56	1.27	14.71	0.00	0.00	0.04
skewness-min	-21.19	-0.62	1.59	0.00	0.00	0.04
skewness-std	0.00	1.60	18.89	0.00	0.01	0.05
symbols-max	0.00	13.09	429.00	0.00	0.00	0.00
symbols-mean	0.00	3.01	41.38	0.00	0.00	0.00
symbols-min	0.00	1.44	12.00	0.00	0.00	0.00
symbols-std	0.00	3.06	107.21	0.00	0.00	0.00
symbols-sum	0.00	71.04	1648.00	0.00	0.00	0.00

Алгоритм работы AutoSklearn

После выбора начальных пайплайнов при помощи метаобучения, происходит следующее:

1. AutoSklearn использует **итеративный процесс поиска** в пространстве моделей и гиперпараметров. На каждом шаге выбирается новый пайплайн, который затем обучается и оценивается на заданных данных.
2. Результаты каждого шага **оцениваются с использованием выбранной метрики** качества (например, точность, F1-мера, среднеквадратичная ошибка).
3. Auto-sklearn использует **байесовский оптимизатор** для настройки гиперпараметров, что позволяет эффективно находить комбинации параметров, улучшающие производительность модели.
4. Если это указано, AutoSklearn **может создать ансамбль из лучших найденных моделей** для повышения обобщающей способности.
5. AutoSklearn возвращает **оптимальный пайплайн машинного обучения**, включая настроенные гиперпараметры и предобработку данных.



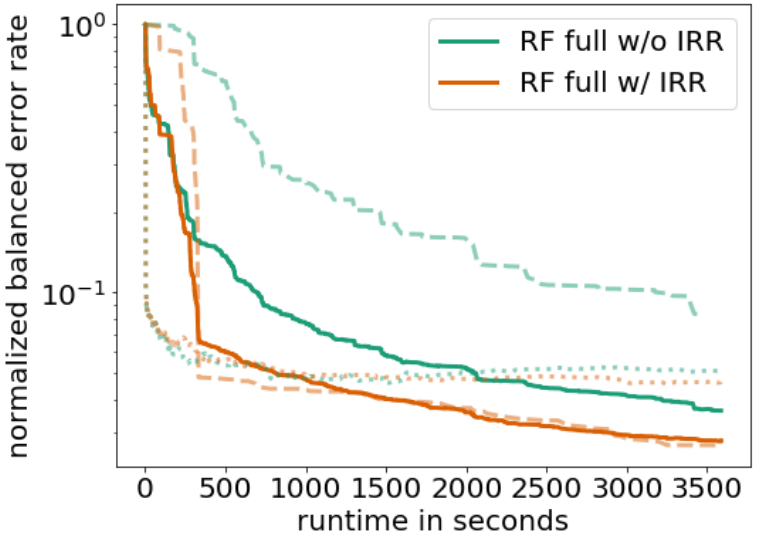
AutoSklearn 2.0 (2020)

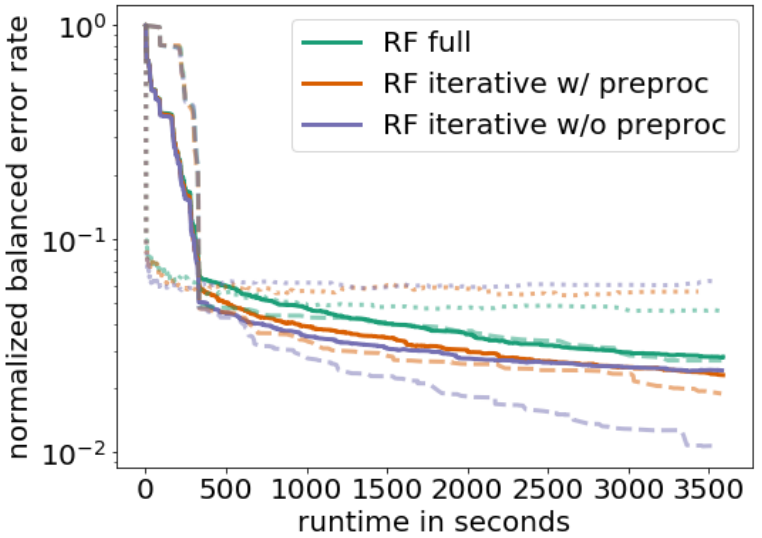
1. В обновленном варианте пакета *каждый pipeline способен совершать раннюю остановку и сохранять результаты промежуточных вычислений*. Это изменение кардинально улучшило производительность и качество работы.

2. Следующим нововведением стало *ограничение множества алгоритмов, в котором производится перебор, до моделей, которые можно обучать итеративно, в частности, методы, основанные на деревьях решений*.

3. Изменился подход к мета-обучению, предыдущая версия библиотеки использовала мета-признаки для определения схожих между собой датасетов. В Auto-sklearn 2.0 реализован другой подход: *создано единое портфолио лучших решений для различных датасетов* (ранее мы выбирали пайплайны, которые лучше всего работают на датасетах, близких к нашему по метапризнакам; в новой версии были собраны пайплайны, которые работают лучше всего на большинстве известных датасетов).

4. Был добавлен *автоматический выбор стратегии подбора наилучшей модели*.





AutoSklearn: практика

https://colab.research.google.com/drive/1E1rbc0AXAX2EFsf-6Dk4BSeXSRlEjcda

LightAutoML (LAMA, 2020)

LightAutoML (LAMA) - фреймворк для автоматического машинного обучения от лаборатории SberAI Lab.

LightAutoML сделан достаточно **легковесным, удобным и эффективным** для решения различных классов задач (бинарной/мультиклассовой классификации и регрессии) на табличных данных, в которых могут содержаться различные типы признаков одновременно: числа, тексты, категориальные данные, даты и другие.

"Существовавшие аналоги отличала известная прожорливость (они забирали много ресурсов) и требовательность к IT-инфраструктуре. LAMA же максимально облегчена для быстрой работы. Мы сами удивились, когда при прогоне OpenML увидели, что каким-то чудом обходим более тяжёлые решения" - цитата от разработчика.

LightAutoML (LAMA, 2020)

LAMA содержит в себе следующие части:

1. **Предобработка данных и генерация признаков**
   Автоматический препроцессинг данных и генерация признаков позволяет экономить время для написания действительно важного кода, основанного на бизнес-знаниях.
2. **Библиотека пресетов для популярных ML-задач и отчеты**
   Сочетает *пресеты, то есть готовые пайплайны*, для популярных задач регрессии, классификации и поддерживает работу с NLP и CV. По итогам прогона LightAutoML на задачах создается подробный отчет по модели.
3. **Модульная архитектура и кастомизация**
   Гибкость фреймворка позволяет писать и использовать собственные абстракции

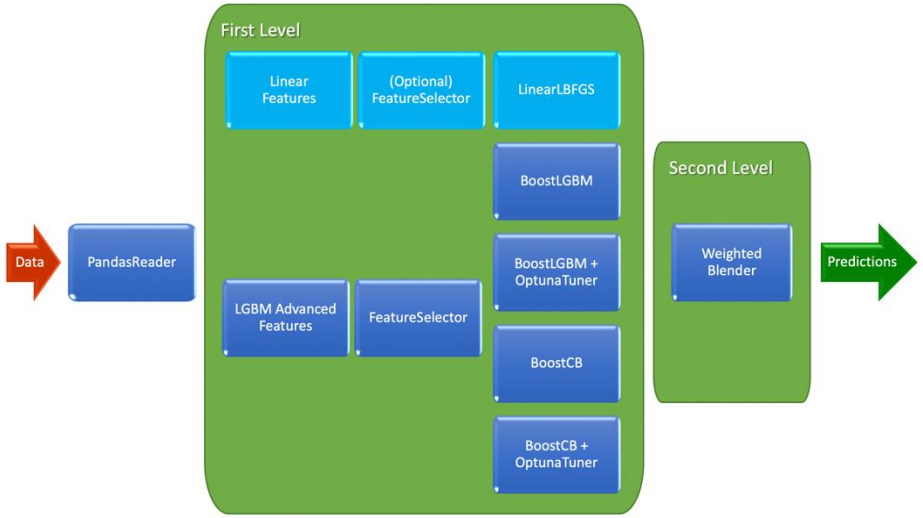
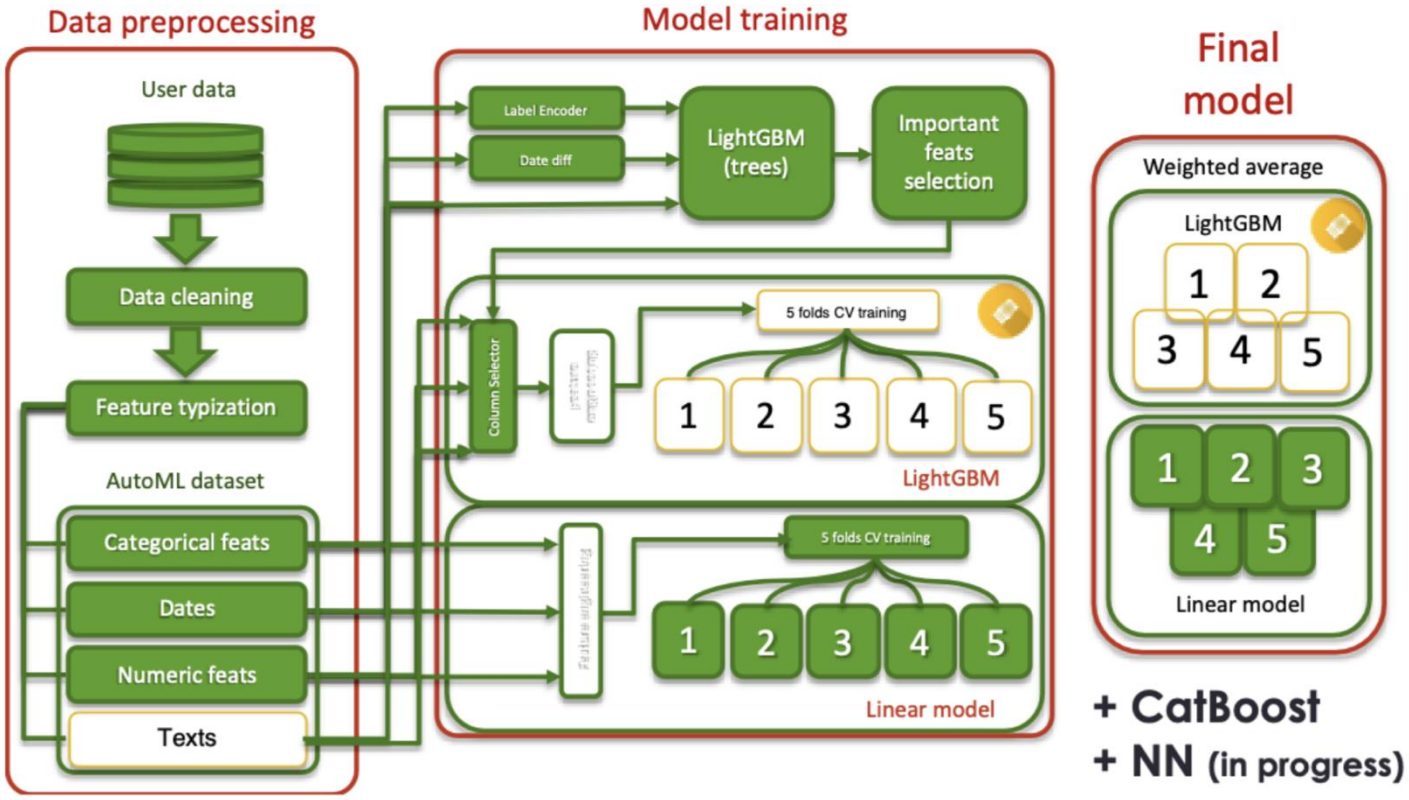


Схема и особенности работы LightAutoML

- Специально чистить данные не нужно — LAMA это хорошо делает сама. Например, обрабатывает пропуски или определяет, имеет ли она дело с категориальной переменной или вещественной.
- На выходе специалист получит не только готовую модель «из коробки», но и результаты всех запусков, которые происходили внутри, а ещё — итоговый отчёт о разработке.
- При этом в соревнованиях результат работы LAMA сравним с 10% лучших участников.
- LAMA работает только с двумя типами моделей - gradient boosted decision trees (GBMs) и линейными моделями, что значительно сокращает время без ущерба для производительности для решаемых типов задач и данных.



Подробная документация по библиотеке от разработчиков здесь.

LAMA

- Подробная документация по библиотеке от разработчиков здесь
- Туториал от разработчиков
- Туториал на платформе Kaggle

Oboe (2019)

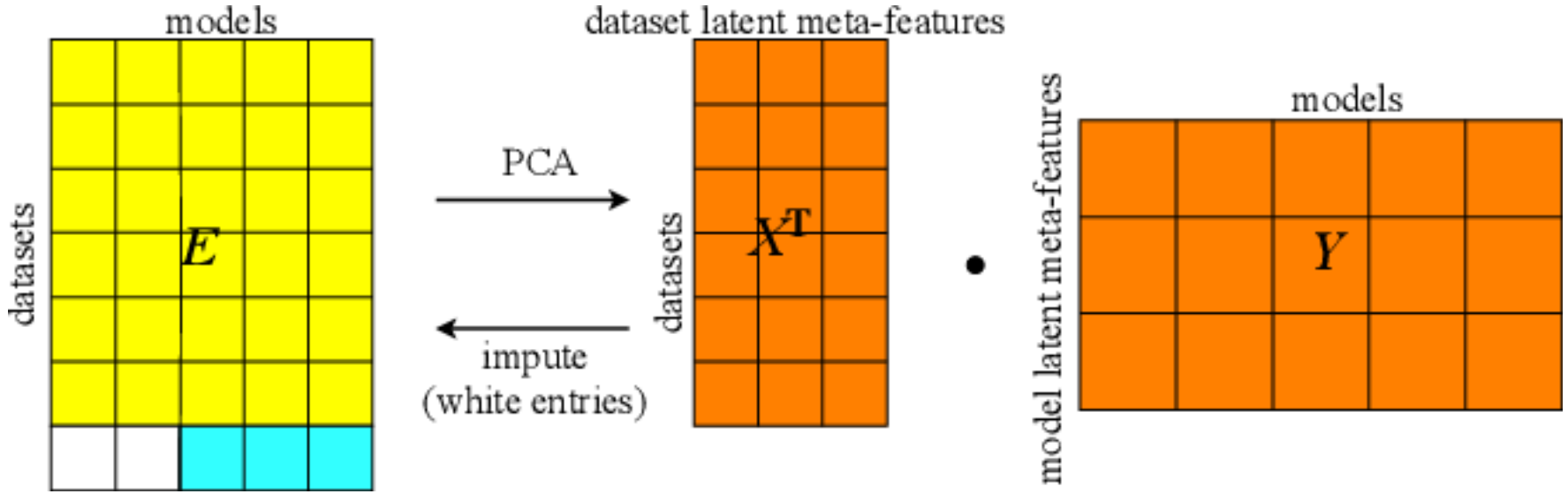
Фреймворк Oboe обеспечивает начальную настройку для AutoML: **система выбирает хороший алгоритм и комбинацию гиперпараметров из дискретного набора вариантов**. Полученную модель можно использовать непосредственно, или гиперпараметры можно дополнительно настраивать.

Название фреймворка происходит от музыкального инструмента oboe: oboe играет начальную ноту в оркестре, от которой другие инструменты начинают настраиваться в процессе подготовки к выступлению.

Алгоритм работы Oboe

Offline-фаза:

- Oboe формирует матрицу ошибок на кросс-валидации, полученных при использовании большого числа моделей обучения с учителем (алгоритмы вместе с гиперпараметрами) на большом числе реальных датасетов.
- Затем Oboe подгоняет модель низкого ранга к этой матрице, чтобы выявить латентные (скрытые) низкоразмерные мета-признаки для моделей и наборов данных. Процедура оптимизации обеспечивает, чтобы эти мета-признаки наилучшим образом предсказывали ошибки на кросс-валидации среди всех линейных моделей.



Алгоритм работы Oboe на новых данных

Шаг 1: Запуск нескольких тестовых моделей

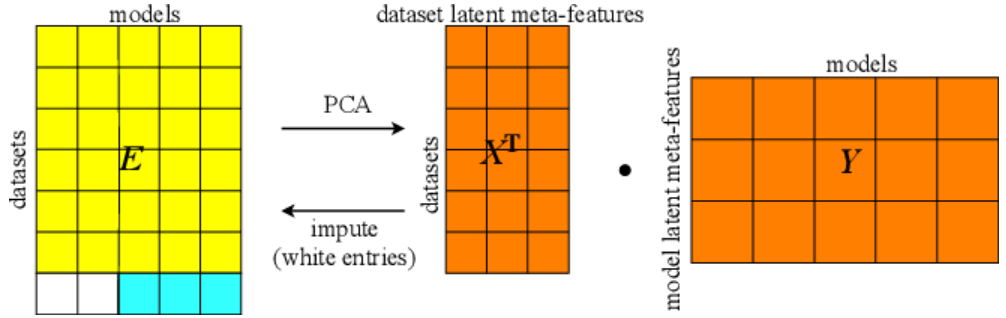
- Oboe не перебирает все модели, а пробует несколько (голубые кубики на картинке).
- Для них вычисляются ошибки на кросс-валидации.

Шаг 2: Встраивание нового датасета в низкоразмерное пространство

- При помощи метода наименьших квадратов (или другого подходящего метода) получаем метапризнаки нового датасета *xnew* так, чтобы произведение $xnew \cdot Y$ давало строку с метриками на кросс-валидации для нового датасета (на иллюстрации это последняя строка в матрице E).

Шаг 3: Предсказание лучших моделей

- Теперь у нас есть метапризнаки нового датасета *xnew.* Oboe ищет похожие датасеты в своей базе (матрице U).
- Он смотрит, какие модели хорошо работали на этих датасетах
- Предлагает лучшие модели для нового датасета **без полного перебора**!



Offline-фаза Oboe

На картинке приведен список моделей и их гиперпараметров, перебираемых в фреймворке Oboe:

Algorithm type	Hyperparameter names (values)
Adaboost	n_estimators (50,100), learning_rate (1.0,1.5,2.0,2.5,3)
Decision tree	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,0.0001,1e-05)
Extra trees	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,0.0001,1e-05), criterion (gini,entropy)
Gradient boosting	learning_rate (0.001,0.01,0.025,0.05,0.1,0.25,0.5), max_depth (3, 6), max_features (null,log2)
Gaussian naive Bayes	-
kNN	n_neighbors (1,3,5,7,9,11,13,15), p (1,2)
Logistic regression	C (0.25,0.5,0.75,1,1.5,2,3,4), solver (liblinear,saga), penalty (l1,l2)
Multilayer perceptron	learning_rate_init (0.0001,0.001,0.01), learning_rate (adaptive), solver (sgd,adam), alpha (0.0001, 0.01)
Perceptron	-
Random forest	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,0.0001,1e-05), criterion (gini,entropy)
Kernel SVM	C (0.125,0.25,0.5,0.75,1,2,4,8,16), kernel (rbf,poly), coef0 (0,10)
Linear SVM	C (0.125,0.25,0.5,0.75,1,2,4,8,16)

TensorOboe (2020)

- Фреймворк Oboe предназначен для поиска оптимальных моделей при помощи матричной факторизации. Для его использования требуется **предобработанный датасет**: закодированные при помощи one-hot encoding категориальные признаки, стандартизация данных (признаки должны иметь нулевое среднее и единичную дисперсию).
  Статья про Oboe (на конференции KDD 2019)

- TensorOboe - обновление фреймворка, при помощи которого происходит поиск оптимальных пайплайнов, которые включают в себя **не только модель, но и предобработку данных**: заполнение пропусков, кодирование, масштабирование данных и снижение размерности.
  Статья про TensorOboe (на конференции KDD 2020)

Offline-фаза TensorOboe

На картинке приведен список моделей и их гиперпараметров, перебираемых в фреймворке Oboe:

Component	Algorithm type	Hyperparameter names (values)
Data imputer	Simple imputer	strategy (mean, median, most_frequent, constant)
Encoder	null	-
	OneHotEncoder	handle_unknown (ignore), sparse (0)
Standardizer	null	-
	StandardScaler	-
Dimensionality reducer	null	-
	PCA	n_components (25%, 50%, 75%)
	VarianceThreshold	-
	SelectKBest	k (25%, 50%, 75%)
Estimator	Adaboost	n_estimators (50,100), learning_rate (1.0,1.5,2.0,2.5,3)
	Decision tree	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,1e-4,1e-5)
	Extra trees	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,1e-4,1e-5), criterion (gini,entropy)
	Gradient boosting	learning_rate (0.001,0.01,0.025,0.05,0.1,0.25,0.5), max_depth (3, 6), max_features (null,log2)
	Gaussian naive Bayes	-
	Perceptron	-
	kNN	n_neighbors (1,3,5,7,9,11,13,15), p (1,2)
	Logistic regression	C (0.25,0.5,0.75,1,1.5,2,3,4), solver (liblinear,saga), penalty (l1,l2)
	Multilayer perceptron	learning_rate_init (1e-4,0.001,0.01), learning_rate (adaptive), solver (sgd,adam), alpha (1e-4, 0.01)
	Random forest	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,1e-4,1e-5), criterion (gini,entropy)
	Linear SVM	C (0.125,0.25,0.5,0.75,1,2,4,8,16)

Oboe: практика

https://colab.research.google.com/drive/10soilSEclI-L4wx1BUsAb8rWK3q4z2SB

Gluon и AutoGluon (2020)

В январе 2020 года компания Amazon выпустила библиотеку с открытым исходным кодом, которая поможет разработчикам быстро создавать приложения с использованием машинного обучения. Проект под названием AutoGluon (Gluon - *Deep Learning API from AWS and Microsoft)* создан в подразделении Amazon Web Services (AWS).

AutoGluon - это надстройка над библиотекой Gluon, комбинирующая в себе мощь и удобство нейронных сетей, а также градиентный бустинг.

Gluon - это библиотека, предназначенная для создания и обучения нейронных сетей. Он разрабатывается совместно компаниями Amazon и Microsoft и предоставляет удобные средства для работы с глубоким обучением. Gluon предоставляет высокоуровневый API для создания нейронных сетей, который более дружелюбен к разработчикам.

AutoGluon (2020)

AutoGluon **осуществляет аккуратную предобработку данных**, а затем применяет **глубинное обучение, бустинги и многослойные ансамблевые методы**. Фреймворк автоматически распознает типы данных, включая текстовые столбцы, и делает для них соответствующую предобработку.

Если обобщить тезисно, то AutoGluon использует следующие подходы:

- комбинация бустингов и нейронных сетей
- стэкинг
- бэггинг

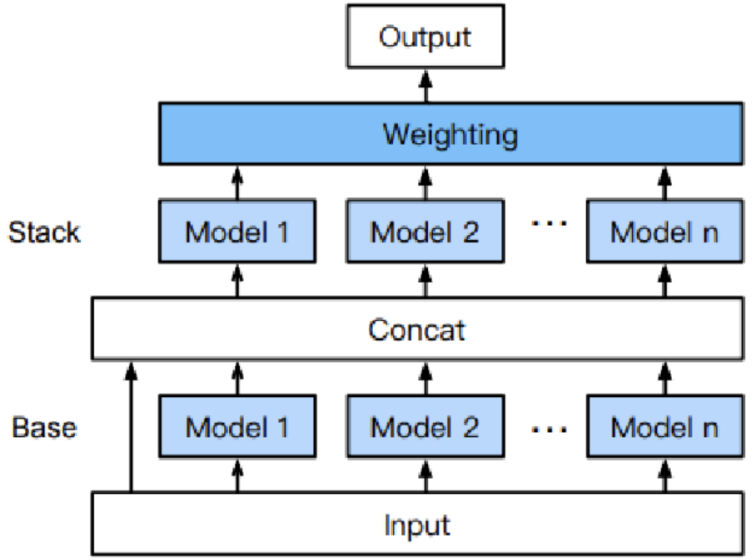


Figure 2. AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and n types of base learners.

AutoGluon: практика

https://colab.research.google.com/drive/1LbyMTtmP8-dPLys2PjWk-w3KLbg7xozv

Недостатки AutoML подхода

AutoML (Автоматическое машинное обучение) подход имеет множество преимуществ при прогнозировании временных рядов, таких как автоматизация процесса подбора моделей и гиперпараметров. Однако у этого подхода есть и свои недостатки. Вот некоторые из них:

- Время и ресурсы, требуемые для перебора всех возможных моделей и гиперпараметров, могут быть слишком велики, что может привести к выбору не оптимальной модели в условиях ограниченного времени
- AutoML инструменты часто функционируют как "черный ящик", к которому сложно применить глубокую интерпретацию и понять причину выбора тех или иных моделей
- Сложные мультивариативные временные ряды или наборы данных с нерегулярными временными метками могут представлять трудности для AutoML
- Человеческое знание и экспертное знание иногда необходимо для глубокой настройки моделей, и эта настройка может быть неудовлетворительной или невозможной с AutoML. Автоматически сгенерированные модели требуют тщательной валидации, чтобы убедиться в их адекватности и применимости
- Огромное количество зависимостей в AutoML-фреймворках делает их нестабильным для использования инструментом