Манифест языка SML – Simple Machine Learning language

Оглавление

Введение

Машинное Обучение и Наука о Данных развиваются в последние годы стремительными темпами и приносят большую пользу обществу. Например, использование искусственных нейронных сетей позволило добиться значительного прогресса в решении таких задач, как распознавание изображений и речи, машинный перевод, позволило оснастить автомобили компьютерным зрением.

Однако, эффективное использование методов машинного обучения для решения повседневных задач промышленности и бизнеса по-прежнему сопряжено с рядом сложностей, главной из которых является высокий порог входа, то есть то количество усилий, которые понадобятся до того, как использование методов даст свои плоды, будь то оптимизация расходов или повышение качества выпускаемой продукции.

Такое положение вещей связано с рядом обстоятельств.

Машинное обучение часто описывается как «область знаний, которая дает компьютерам возможность решать задачи без явного программирования». Однако опытные специалисты знают, что разработка эффективных методов машинного обучения часто является утомительным занятием, требуя значительного опыта работы с алгоритмами, экспертных знаний в предметной области, трудоемкого подбора моделей и их параметров методом грубой силы.

Таким образом, вопреки убеждениям энтузиастов, машинное обучение все еще требует значительного явного программирования. Это делает невозможным использование методов машинного обучения экспертами предметных областей, не являющихся специалистами в области программирования.

В результате, для построения эффективной цепочки анализа данных требуется участие большого числа специалистов, начиная от инженеров, управляющих хранилищем данных, программистов, реализующих рутинные операции над данными, и заканчивая штатом ученых и экспертов в области анализа данных, которые непосредственно отвечают за получение

результата. Всё это приводит к значительным организационным и финансовым затратам.

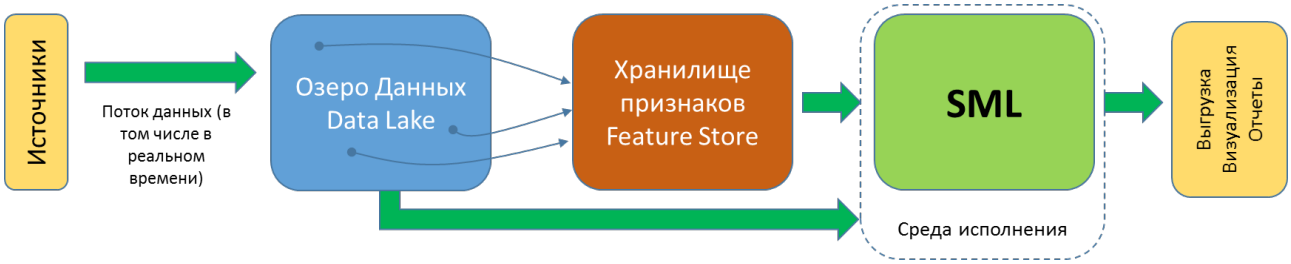
Еще один аспект связан с тем, что ручной подбор моделей и их параметров может привести к неоптимальным решениям, а многие из используемых подходов и методов представляют собой сложные «черные ящики», которые трудно интерпретировать, понять, отладить, и которым трудно доверять впоследствии.

Таким образом, мы наблюдаем постоянно растущий спрос на системы машинного обучения, которые могут использоваться «из коробки» неспециалистами. Чтобы быть эффективными на практике, такие системы должны автоматически выбирать алгоритм для решения той или иной задачи, сформулированной в терминах человеческого языка, а не языка программирования, а также эффективно настраиваться на имеющиеся данные, обучая модели и подбирая соответствующие гиперпараметры.

Компания Открытые Технологии предлагает сообществу разработчиков объединить усилия в разработке эффективной, свободно распространяемой системы, ядром которой будет являться специализированный язык ***(SML – Simple Machine Learning language)****, дающий возможность экспертам предметных областей, не являющихся программистами, решать разнообразные задачи, возникающие в их профессиональных областях, методами машинного обучения, не погружаясь в специфику науки о данных.*

Система должна быть построена на принципах автоматического машинного обучения (AutoML) и должна выполнять следующие основные функции:

- Загрузка и хранение данных, формирование озера данных (Data Lake).
- Первичная нормализация данных и извлечение признаков (Feature Store).
- Автоматизация выбора и преобразования признаков в форму пригодную для использования совместно с методами машинного обучения.
- Автоматизация работы с неполными данными и выбросами в них.
- Поиск оптимальной модели из большого набора подходящих для решения той или иной задачи с учетом имеющегося набора данных.
- Автоматизированный подбор гиперпараметров модели.
- Обучение модели, избегая пере- или недо- обучения, адаптация к потоку данных.
- Интерпретацию полученных результатов, желательно на человеческом языке в диалоге с пользователем.
- Надежная предсказуемая реализация, эффективная с точки зрения ограничений на время вычислений, память, объем данных и другие ресурсы.



Озеро данных (Data Lake) может быть организовано стандартными средствами, например, для этого прекрасно подходит кластер Hadoop, загрузка данных может осуществляться одним из средств из богатой экосистемы этого решения.

Хранилище признаков (Feature Store) позволяет централизовать знание о имеющихся данных, переиспользуя результаты работы специалистов, уже получивших полезные признаки из имеющихся ненормализованных данных.

Оба этих слоя не являются предметом рассмотрения данного документа. Здесь же мы остановимся на нашем видении концепции языка SML, а также его среды исполнения.

SML - Simple Machine Learning language

Назначение языка

Центральная концепция системы – язык описания задач предметной области Simple Machine Learning language, посредством которого пользователи, не являющиеся специалистами в машинном обучении получают доступ ко всей мощи концепции автоматического машинного обучения (AutoML).

Целевая аудитория языка – специалисты в предметной области, не являющиеся математиками и датасайтистами, т.е. люди, хорошо понимающие в том, с чем они работают каждый день: технологи, главные инженеры, диспетчера, конструктора, логистики.

Язык является текстовым, то есть принципиально отделимым от графических систем проектирования. На начальном этапе это позволит сэкономит ресурсы разработчиков, однако при необходимости и достаточной зрелости языка не будет сложным разработать соответствующие графические средства разработки. При этом текстовое представление должно быть эргономичным и лаконичным, без лишних синтаксических конструкций, должно позволять быструю разработку и прототипирование.

Центральной концепцией языка является то, что для самого неподготовленного в области анализа данных пользователя, он дает возможность, используя максимально высокоуровневые функции, в принципе, решить любую задачу из конкретной профессиональной области, пусть и в пределах некоторых шаблонов и перенастроенных паттернов. При этом не требуется программирование на языках высокого уровня, не требуются какие-либо особые знания в области машинного обучения и анализа данных. Пользователю надо иметь возможность изложить постановку задачи на том языке, на котором он способен изложить, на бытовом языке, например, на уровне алгоритма вида:

- взять данные,
- увидеть паттерн события,
- если паттерн постоянный, то сделать А,
- если паттерн изменился, то сделать Б.

Назовем такие высокоуровневые функции **Функциональными Блоками**.

Функциональные Блоки

Функциональные Блоки реализуют высокоуровневые задачи, что является о принципиально важным. Мы не погружаемся в глубины методов машинного обучения, а даем пользователю «кирпичики» вида «спрогнозируй», «обнаружь

аномалию», «рассчитай по такой то формуле», «сделай группировку», «найди разное», «найди общее», «выяви паттерн подобного рода», «синхронизируй», «сделай голосование», однако для подготовленных пользователей могут существовать и блоки более низкого уровня, такие как «классифицируй», «примени регрессию», «проверь статистическую гипотезу». Отдельный тип Блоков отвечает за загрузку данных из Озера Данных или Хранилища Признаков.

Таким образом, важной особенностью языка должно являться то, что он должен обладать слоистой сложностью – любые задачи должна быть возможность решать полностью на трех уровнях сложности с точки зрения языка в соответствии с подготовкой пользователя в области анализа данных:

- **Beginner** – оперирует базовым набором высокоуровневых Блоков без необходимости настройки или кастомизации их параметров.
- **Middle** – оперирует расширенным набором блоков, включая низкоуровневые блоки машинного обучения, с возможностью задавать параметры по умолчанию, выбирать используемые модели и т.д.
- **Master** – имеет возможность разрабатывать собственные Блоки на языках программирования высокого уровня из предлагаемого набора (Python, Java, Scala) используя предлагаемы системой API.

Для каждой целевой предметной области должен существовать набор Блоков, позволяющих решать потенциально произвольные задачи на самом простом уровне сложности. В процессе решения, однако, не запрещается использование блоков из разных уровней, однако в базовом воплощении задача должна иметь возможность решения блоками уровня Beginner.

Граф блоков (пайплайн) и унифицированный формат данных

Еще одним важным средством языка является возможность объединения блоков в цепочки для получения **Графа Блоков**, когда результат работы одного блока отправляется на вход одного или нескольких других, или, наоборот, результаты работы нескольких блоков объединяются. При этом в языке должны быть предусмотрены средства для организации циклов, ветвлений, обработки исключений, синхронизации как по времени исполнения, так и по используемым участкам данных, проходящих обработку.

Между блоками производится обмен данными в унифицированном формате, при этом, по возможности, данные не копируются и не передаются по сети, а передаются указатели (дескрипторы) на данные в едином кластерном оперативном хранилище. Сами данные представляют собой матрицу исходных и целевых признаков в табличном формате с именованными колонками.

Возможно, появится необходимость использования нескольких форматов, например, одного формата для временных рядов, другого для обычных многомерных объектов, однако принципиально что количество таких форматов ограничено и пользователю нет необходимости разбираться что в каком формате передается и как такие форматы между собой преобразовывать.

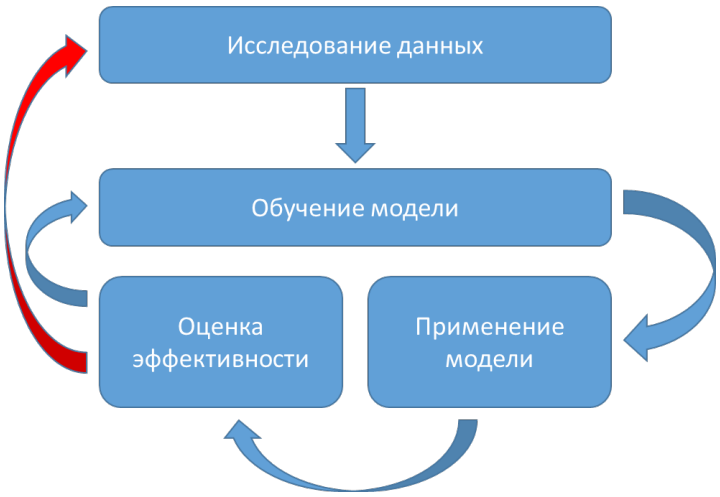
Приведение данных из Озера Данных к универсальному формату осуществляется системой (средой исполнения). В языке это реализуется блоком загрузки данных. Помимо преобразования форматов такой блок автоматически осуществляет такие операции как заполнение пустот и отсечение выбросов.

Жизненный цикл блоков

Важнейшей концепцией языка является жизненный цикл блока, то есть особенности его исполнения средой и взаимодействия с пользователем. При этом жизненный цикл реализуется средой исполнения автоматически и не требует дополнительного написания какого-либо кода.

В жизненном цикле бока разделяют 4 фазы:

- **Фаза исследования данных.** На этом этапе блок получает на вход пример входных данных и осуществляет предварительную настройку на них. Пользователь имеет возможность быстро оценить приспособленность блока в настройками по умолчанию к его набору данных и, при необходимости, провести тонкую настройку блока на самом высоком уровне (не опускаясь до настройки отдельных гиперпараметров). При этом в среде исполнения возможно ознакомиться как непосредственно с результатами работы блока, так и с оценкой качества работы. Как только результат работы блока устроит пользователя – система переходит ко второй фазе жизненного цикла блоки.
  На данном этапе «под капотом» происходит загрузка данных, выбор метода нормализации и отбора признаков, выбор одной из множества подходящих для решения задачи моделей машинного обучения, осуществляется обучение с кросс-валидацией на исходной выборке, и автоматическая настройка гиперпараметров модели.
- **Фаза обучения моделей.** На этом этапе происходит обучение моделей блока на динамически загружаемой в систему выборке данных, при этом используются настройки параметров блока, полученных на первом этапе.
- **Фаза исполнения.** На этом этапе происходит исполнение моделей блока для получения тех результатов, для которых блок предназначен. Например, блок прогнозирования временного ряда выдает прогноз на определенное количество отсчетов в будущее.
- **Фаза оценки качества работы.** На четвертом этапе блок автоматически проверяет качество своей работы, например, качество прогнозирования, сравнивая результат работы с поступающими в систему данными. Если качество работы перестает отвечать заданным критериям – система переходит или к фазе обучения (на «свежей» порции данных), или, если данные существенным образом поменялись, на этап исследования данных для выбора иных параметров блока или вообще его замены на более подходящий к изменившимся обстоятельствам.



Таким образом, система динамически адаптируется к потоку данных, при этом пользователю нет необходимости писать для этого специальный код. Фактически вместо написания кода происходит настройка готовых блоков, «под капотом» которых скрыты мощные методы автоматического машинного обучения (AutoML).

Заключение

Язык SML позволит (на самом высоком уровне) полностью избежать таких трудоемких рутинных операций, как загрузка, нормализация и очистка данных, отбор признаков, выбор модели и ее гиперпараметров, преобразование форматов и настройка взаимодействия между разными библиотеками машинного обучения. Всё это требует большого количества времени и серьезных знаний в области анализа данных. В результате язык даст возможность экспертам, не являющимся специалистам в области машинного обучения, просто решать задачи в своих профессиональных областях деятельности.

Среда исполнения обеспечит легкость отладки и прозрачную интерпретируемость моделей.

Компания Открытые Технологии надеется, что данной декларацией ей удастся привлечь широкий круг специалистов из сообщества разработки программного обеспечения с открытым исходным кодом к разработке спецификации языка SML и среды его исполнения.