В результате применения обычного градиентного спуска в среднем на одну эпоху тратилось в среднем 93 мс, при этом точность предсказания на тестовой выборке составила 0.32. Из этого видно, что 10 эпох не достаточно для достижения высокой точности. При использовании пакетного градиентного спуска с размером пакета 128 элементов среднее время обучения одной эпохи составило 153 мс, качество обучения – 0.78. Для стохастического градиентного спуска на обучение одной эпохи потребовалось 61 с, качество обучения составило 0.84. Для обучения использовался шаг равный 0.5. Для оценки качеств модели использовалась метрика *accuracy* [4].

Из полученных результатов можно сделать вывод о том, что обычный градиентный спуск работает гораздо медленнее при достижении минимальной ошибки в сравнении со стохастическим и пакетным градиентный спуском, однако он значительно лучше минимизирует функцию. Стохастический градиентный спуск лучше показывает себя на задачах, в которых для обучения требуется небольшое количество данных, для более сложных задач с большим набором данных в обучающей выборке достижение минимума может также оказаться долгим процессом. Для больших выборок подходит пакетный градиентный спуск, он быстрее в сравнении с обычным и в то же время быстрее сходится к минимуму в сравнении со стохастическим. Таким образом, в случае, когда достаточно найти значение близкое к минимуму за приемлемое время, имеет смысл использовать стохастический или пакетный градиентный спуск в зависимости от решаемой задачи.

Список использованных источников:

1. Набор данных notMNIST [Электронный ресурс] – Режим доступа: http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html
2. Функция relu [Электронный ресурс] – Режим доступа: https://arxiv.org/pdf/1803.08375.pdf
3. Функция softmax [Электронный ресурс] – Режим доступа: https://arxiv.org/pdf/1811.03378.pdf
4. Определение метрики accuracy в математической статистике [Электронный ресурс] – Режим доступа: http://gsp.humboldt.edu/OLM_2015/Courses/GSP_216_Online/lesson6-2/metrics.html

ОБЛАЧНАЯ ИНФРАСТРУКТУРА КАК КОД НА ПРИМЕРЕ СИСТЕМЫ ДЛЯ АВТОМАТИЗАЦИИ КОНТРОЛЯ ДОСТУПА К SAP

Киселёв Д.О.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Егорова Н.Г. – канд. техн. наук,
доцент каф. Информатики

В настоящее время использование облачных сервисов стало стандартом при разработке программного обеспечения. Будь это маленький сайт или большая промышленная система. Потребности в мощности ресурсов очень динамические: сегодня систему использует сто человек, через полгода - тысяча, следовательно, становится вопрос простого перехода из одной инфраструктуры на другую. В случае использования облачных сервисов и подхода «инфраструктура как код», данный процесс максимально упрощается.

Инфраструктура как код (англ. Infrastructure as code (IaC)) – это способ создания и управления ресурсами для центров хранения и обработки данных методом их описания в исходном коде. Данный подход приходит на замену настройке оборудования вручную или с помощью интерактивных инструментов. Для описания инфраструктуры чаще всего используется декларативный язык, так же оно может находится в системе контроля версий. Подходы IaC продвигаются для облачных вычислений, которые иногда называют инфраструктура как сервис (IaaS).

Преимущества IaC можно описать в трех направлениях: стоимость, скорость, риски. Снижение стоимости происходит как за счет более рационального использования вычислительных ресурсов, так и за счет снижения трудозатрат на их обслуживание, позволяя больше сфокусироваться на решении проблем бизнеса. Автоматизация инфраструктуры обеспечивает скорость за счет более быстрого выполнения при настройке инфраструктуры. Автоматизация убирает риск, связанный с человеческой ошибкой, что позволяет уменьшить время простоя и повысить надежность. Эти преимущества помогают корпоративному сегменту двигаться в направлении развития практик DevOps.

Архитектура системы для автоматизации и контроля доступа к SAP построена с использованием большого количества облачных сервисов, причем разных поставщиков. Изначально единственным способом создания нового окружения было исполнения шагов, описанных в текстовом документе вручную. Для того, чтобы развернуть новое окружение, тратилось очень много времени. Так же этот процесс был подвергнут ошибкам из-за разночтения инструкции. Соответственно стал вопрос об автоматизации развертывания инфраструктуры. Схема архитектуры представлена на рисунке 1.

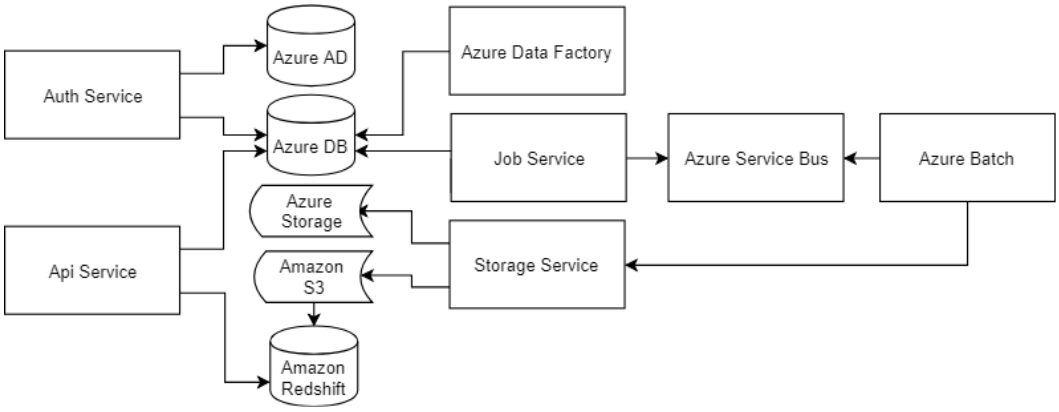

Рис. 1 – Архитектура системы для автоматизации контроля доступа к SAP

Решением проблемы стало использование программного инструмента Terraform от компании HashiCorp. Terraform предоставляет библиотеки для управления большинством сервисов всех крупных облачных поставщиков. Для достижения целей были использованы библиотеки для облаков Azure и AWS. Данное решение помогло описать используемые ресурсы в декларативном стиле, и, в последствии, использовать для создания новых окружений без особых сложностей. Описание каждого ресурса создавалось в отдельном файле на языке конфигураций HCL (Hashicorp Configuration Language). Так же была использована система контроля версий Git, что позволяет видеть изменения в инфраструктуре.

Список использованных источников:

1. Infrastructure as code [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Infrastructure_as_code – Дата доступа: 24.03.2019.
2. Terraform Documentation [Электронный ресурс]. – Режим доступа: https://www.terraform.io/docs/index.html – Дата доступа: 24.03.2019.
3. Terraform on Azure [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/azure/terraform/ – Дата доступа: 24.03.2019.

ИСПОЛЬЗОВАНИЕ АЛГОРИТМА КЛАСТЕРИЗАЦИИ DBSCAN ДЛЯ ФИЛЬТРАЦИИ ВЫБРОСОВ В ДАННЫХ

Ковалёв С.П.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Калугина М.А. – к. ф.-м. н., доцент

При обработке больших объемов данных часто используется такой метод машинного обучения как кластеризация, который помогает найти группы похожих объектов в наборе данных. Но для успешной кластеризации набора данных его необходимо предварительно обработать – очистить данные от выбросов. Одним из таких способов является использование способности алгоритма DBSCAN определять выбросы в наборе кластеризуемых данных.

Кластеризация относится к методам обучения «без учителя». Эти методы не предполагают, что существует правильный ответ, для получения которого можно обучить алгоритм. Кластеризация призвана систематизировать данные и найти группы похожих объектов.

Методы кластерного анализа используются в психологии, археологии, биологии. В настоящее время кластеризация также активно применяется в социологии, экономике, статистике, в исторических исследованиях и многих других сферах. Страховые компании могут разделять автомобили по группам риска, а затем назначать каждой группе различную стоимость страховки. В вычислительной биологии этот метод активно применяется для обнаружения групп генов, демонстрирующих сходное поведение. Это может служить указанием на то, что они будут одинаково реагировать на лечение или что образовались на одном пути биологического развития [1]. Особенно расширилось использование этого метода машинного обучения в связи с появлением и развитием ЭВМ, поскольку это связано с трудоемкостью обработки больших объемов информации.

Алгоритмы кластеризации можно рассматривать как иерархические и неиерархические. Результатом работы иерархических алгоритмов является разбиение данных по уровням. В этом случае исследователь может выбрать любую ступень построенной иерархии при интерпретации