Модуль 8. Эксплуатация моделей машинного обучения.

# Образовательный результат:

Термин “эксплуатация” в технологических проектах означает “использование”, “применение”. Процесс эксплуатации любой программной системы является не только закономерным итогом всего проекта разработки программного обеспечения, но и его наиболее практически полезной частью. Программное обеспечение разрабатывается для того, чтобы кто-то его использовал. Однако после вывода решения в эксплуатацию работа в проекте не завершается, поскольку процесс эксплуатации выявляет ошибки в системе, выдвигает новые задачи перед разработчиками, дает новые идеи. И если процесс эксплуатации выстроен неэффективно, то будут нарастать проблемы с качеством, увеличиваться технический долг по проекту, ресурсы проекта будут расходоваться неоптимально. В итоге неэффективная эксплуатация может привести к неуспешному завершению даже самый качественный проект. Для повышения эффективности эксплуатации создают специальные инструменты, решающие основные эксплуатационные задачи: тестирование, мониторинг, управление, анализ, эксперименты, обнаружение инцидентов. Знание и эффективное владение инструментами для эксплуатации, умение их правильно и быстро установить и настроить для работы, являются востребованными навыками специалистов DevOps/MLOps.

После изучения этого модуля Вы научитесь пользоваться инструментами для эксплуатации программных систем, которые применимы и для проектов машинного обучения.

# В этом модуле:

Успешная эксплуатация программных систем зависит от навыков специалистов команды, обеспечивающей работу программной системы в производственной среде, а также от используемого инструментария, помогающего решать эксплуатационные задачи.

Организации и управлению команд специалистов, занимающихся эксплуатацией программных систем, посвящено большое количество специализированной литературы. Есть специальные системы, помогающие организовать этот процесс эффективно, например “Help Desk”, “Service Desk”, CRM и другие подобные. Это важная часть проекта, требующая отдельного изучения и важная для практического применения, но в данном курсе эта тема не рассматривается, поскольку основная цель курса это познакомить вас с задачами и инструментами DevOps/MLOps. Поэтому далее в этом модуле мы рассмотрим инструменты для:

* мониторинга ресурсов
  + оперативная память,
  + процессорная мощность,
  + жесткий диск,
  + сетевые интерфейсы
* мониторинга процессов
  + проверка качества данных,
  + обработка данных,
  + подбор гиперпараметров,
  + обучение модели,
  + инференс (использование в производственной среде) модели
* контроля системы
  + управление пользователями и группами,
  + управление процессами,
  + управление сетевыми интерфейсами,
* организация взаимодействия отдельных элементов
  + передача файлов,
  + удаленное подключение,
  + настройка защищенного доступа.

Важно понимать, что для задач эксплуатации актуально все, что мы уже обсуждали в предыдущих модулях: автоматизация, унификация, управление версиями используемых объектов, всесторонний контроль и аналитика. Владение инструментами для решения этих задач позволяет сделать процесс эксплуатации более управляемым и эффективным, а работу модели машинного обучения более качественной и надежной. Также это позволяет своевременно выявлять проблемы в работе модели машинного обучения и системы в целом, и принимать меры для устранения неисправностей.

В этом модуле описаны различные инструменты для решения описанных выше задач. Темы, изучаемые в модуле:

1. Средства для мониторинга и управления ресурсами в linux.
2. Утилита Tensorboard для мониторинга.
3. Визуализация и контроль с использованием Grafana.
4. Практический пример использования Grafana для мониторинга проекта машинного обучения.

# Модуль 8. Юнит 1. Утилиты мониторинга и управления ресурсами в linux.

*Введение:*

Linux является очень популярной операционной системой, в том числе для разработки и эксплуатации моделей машинного обучения. Это связано с тем, что большинство библиотек и утилит, использующихся в проектах машинного обучения, эффективнее работают в linux системах, чем в других. С момента своего создания linux был бесплатным программным обеспечением с открытым программным кодом, что предопределило его огромный успех и массовое использование в различных проектах. Любой специалист в любой точке мира мог без технических проблем и финансовых вложений начать использовать linux в своих проектах. В том числе это сделало возможным привлечение большого сообщества разработчиков и инженеров для разработки и эксплуатации программного обеспечения для linux. Все это в полной мере относится и к машинному обучению. Появление большого количества библиотек и фреймворков машинного обучения с открытым программным кодом (jupyter, numpy, sklearn, Tensorflow, PyTorch и многих других) стало возможным с использованием идеологии open-source. Оптимизация выполнения вычислительных операций, благодаря которой библиотеки машинного обучения стали такими эффективными, оказалась возможна благодаря открытости операционной системы linux и использованию “коллективного разума” большого сообщества разработчиков, которые занимались разработкой и отладкой кода, тестированием и применением разработок. **Это позволило создать высокопроизводительные и надежные инструменты машинного обучения, которые активно используются специалистами во всем мире, в том числе и в высоконагруженных системах и в системах с повышенными требованиями к надежности, скорости работы.**

Другим существенным преимуществом использования linux в проектах машинного обучения является простота установки и настройки инструментария. Большинство программ можно установить с использованием всего нескольких команд. Почти все инструменты готовы к использованию сразу после установки с настройками по умолчанию. Простота процедуры начала работы с инструментом и удобство его использования, наличие широкой информационной базы примеров и сообщества специалистов, помогающих в поиске ответов на возникающие вопросы, предопределяют популярность любого инструмента. **Все это привело к тому, что операционные системы linux и программное обеспечение на их основе стали стандартом “де-факто” в проектах машинного обучения.**

В связи с такой популярностью linux от специалистов MLOps требуется хорошее знание этой операционной системы, умение ее администрировать. Без этих навыков невозможно установить, оптимально настроить и использовать большинство инструментов MLOps, которые вы изучили в предыдущих модулях. Вы изучали linux в курсе «Операционная система linux» (и в рамках мастер-класса во втором семестре), при необходимости рекомендуется вспомнить сведения, полученные в этом курсе. Не будет лишним, если вы прочитаете дополнительно одну из многих книг про администрирование linux и подкрепите полученные сведения практической работой в командной строке linux системы, поскольку в основном для работы используется командная строка. Некоторые из книг вы можете найти [по ссылкам в конце модуля](https://tproger.ru/books/linux/).

Основное содержание юнита составляют описания утилит для мониторинга и управления, входящие в состав linux систем. Вы узнаете как получить информацию об используемых системных ресурсах: оперативной памяти, жестком диске, процессорах, сетевых интерфейсах. Также полезными являются сведения о пользователях, группах и сервисах. И, наконец, в юните рассказывается о различных типовых инструментах, используемых системными администраторами и инженерами DevOps/MLOps в повседневной работе: текстовых редакторах, управлении сервисами, запуске по расписанию, удаленной работе.

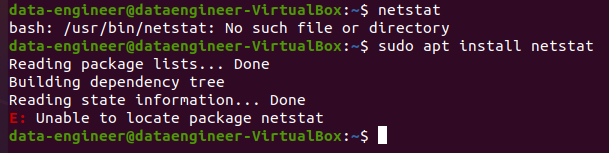
*Содержание юнита:*

Большинство служебных утилит, которые используются для администрирования операционной системы linux и решения практических задач, выглядят одинаково для различных версий linux (RedHat, Fedora, Ubuntu, Kali, CentOS и др.). Тем не менее иногда имеются различия в синтаксисе, например, могут различаться штатные установщики пакетов программного обеспечения в версиях linux:

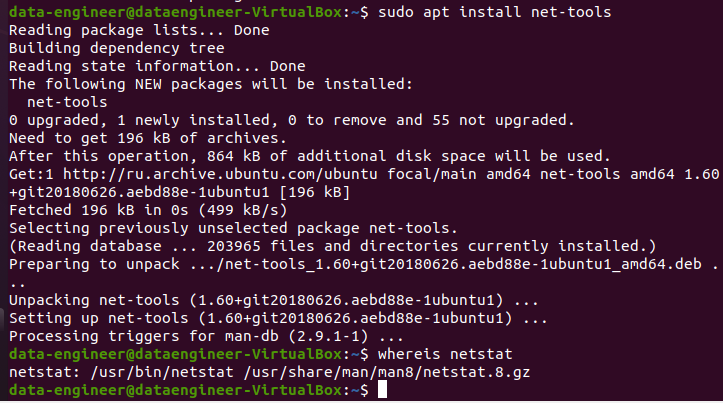
* rpm в RedHat,
* apt, snap или dpkg в Ubuntu и Kali,
* apk в Alpine,
* dnf в Fedora
* yum в CentOS.

Если необходимый вам инструмент не входит в состав операционной системы, то он как правило легко устанавливается. В большинстве случаев достаточно выполнить единственную команду с использованием штатного установщика, либо точно следовать инструкции по установке на официальном сайте разработчика утилиты.

Например, утилита для мониторинга работы сетевого оборудования netstat по умолчанию не входит в набор утилит операционной системы, поэтому необходимо ее установить отдельно.



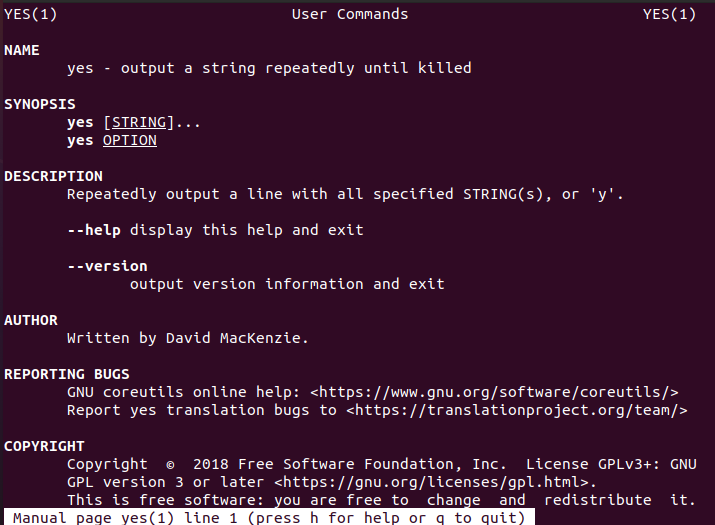
Команда установки неуспешна, так как netstat не устанавливается отдельно, а входит в пакет программ net-tools, именно его и надо установить



С помощью утилиты whereis мы проверили, что установка успешно осуществилась и утилита netstat доступна по адресу /usr/bin/netstat. Утилит в linux очень много, для каждой задачи уже создана специальная утилита. Удобство их использования заключается еще и в том, что в linux для инструментов существует подробная документация, это было заложено в идеологии linux и поддерживается с развитием новых инструментов. Для доступа к технологической документации можно использовать ключ “-h” для любой команды или воспользоваться утилитой man. Например, вот так можно получить справку по команде yes



в результате вы увидите документацию, описывающую применение команды yes.



Документация man содержит описания всех ключей и опций использования для команд linux, ей удобно пользоваться.

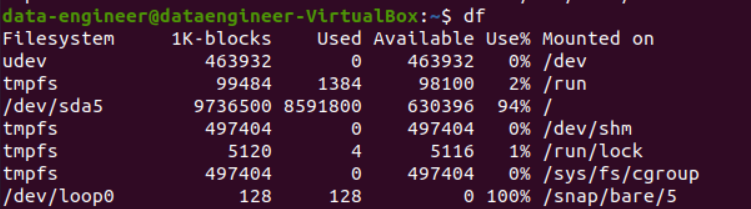
**Давайте рассмотрим некоторые полезные утилиты, которые могут пригодится вам при развертывании и эксплуатации сложных программных систем в операционной системе linux. Также рассмотрим некоторые примеры задач DevOps/MLOps, в которых применяются эти утилиты.**

1. **Утилиты для мониторинга ресурсов**

В проектах машинного обучения активно используются системные аппаратные ресурсы: память на жестких дисках, оперативную память, вычислительную мощность процессоров. При отсутствии должного контроля или при ошибках в программном коде можно получить ситуацию, когда ресурсы заканчиваются. Обычно одними и теми же ресурсами пользуются многие участники команды и сбой одной программы может привести к невозможности работы всей команды. По этой причине необходимо уметь диагностировать задействованные ресурсы, находить опасные тенденции и причины их возникновения.

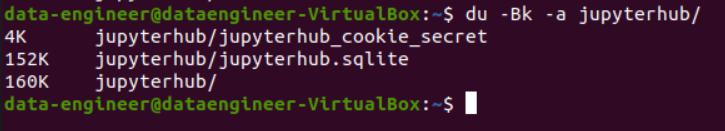
Давайте рассмотрим утилиты для оценки задействования памяти на жестких дисках.

**df**

****

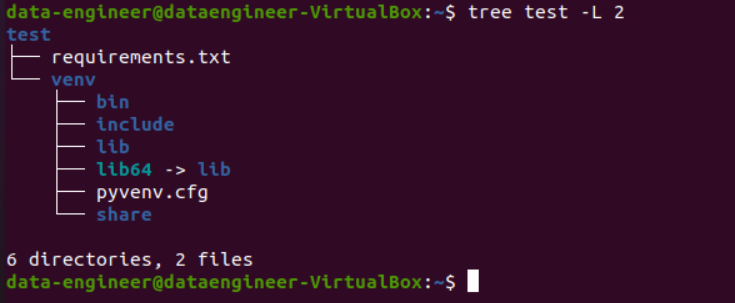
При выполнении этой команды видим объем свободной и задействованной памяти.

**du**

****

Более продвинутой версией утилиты du является утилита **ncdu**, которую надо специально установить, поскольку она не входит в стандартный пакет программ. ncdu обладает простым графическим интерфейсом, в котором можно перемещаться по папкам с использованием клавиш курсора, быстро видеть информацию по размеру папок и даже удалять ненужные папки прямо из этого графического интерфейса.

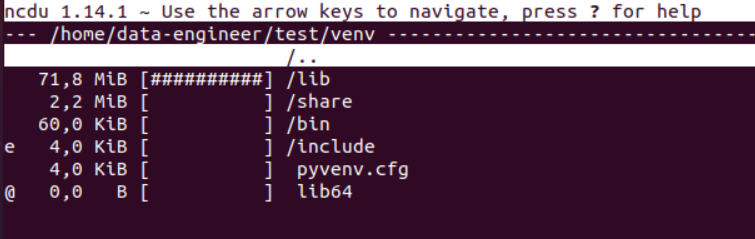
Давайте рассмотрим практическую задачу. Пусть у нас есть папка /test, имеющая структуру для использования виртуального окружения venv следующего вида



Вызов ncdu для просмотра папки /test позволит оценить размер использованного пространства жесткого диска как для папок, так и для отдельных файлов

****

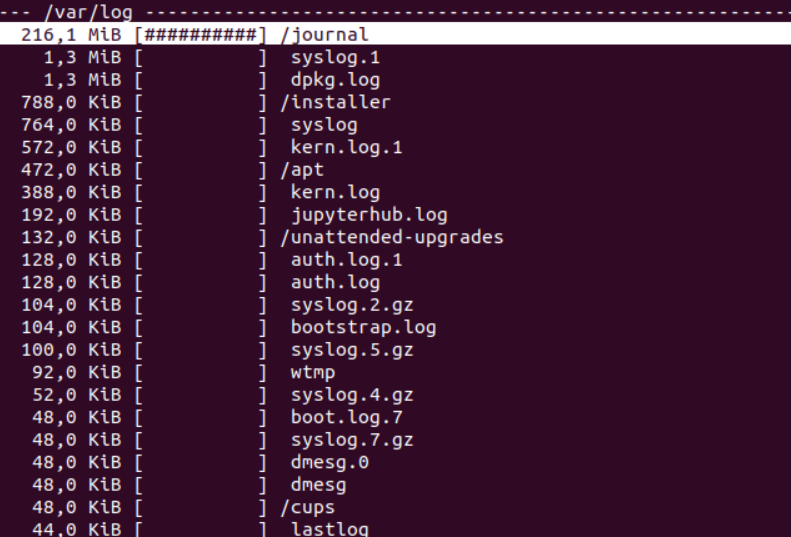
При необходимости можно удобно перемещаться по папкам, детализируя информацию о задействовании пространства жесткого диска внутри папок. Например, вы можете зайти в папку venv и оценить размер ее подпапок.



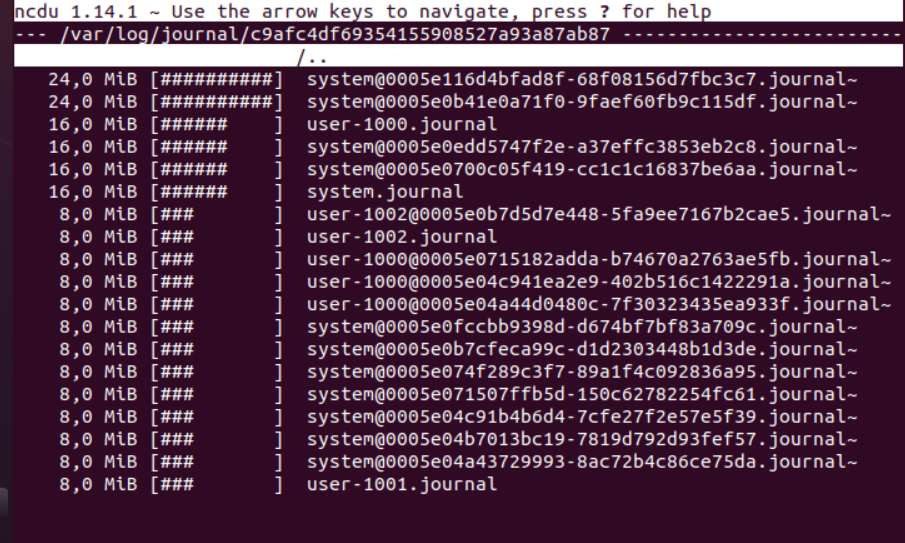
Одно из полезных практических применений этой утилиты: обнаружение разросшихся файлов логов. По умолчанию системы пишут информацию о своей работе в log файлы, обычное место хранения этой информации /var/log. При отсутствии контроля, несмотря на то что log файлы имеют текстовую структуру, размер этой папки может быстро увеличиваться и в итоге привести к нехватке рабочего пространства на жестких дисках. Продиагностировать ситуацию может команда



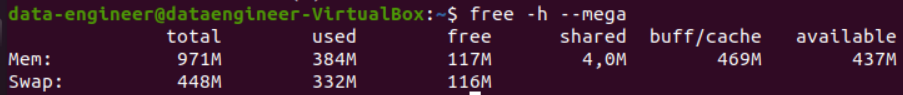
в результате выполнения которой вы увидите информацию о размерах папок внутри /var/log



Видим, что наибольшее пространство занимает /var/log/journal, можем переместиться в него и посмотреть более детально что за файлы занимают это пространство, при необходимости некоторые из них можно удалить.



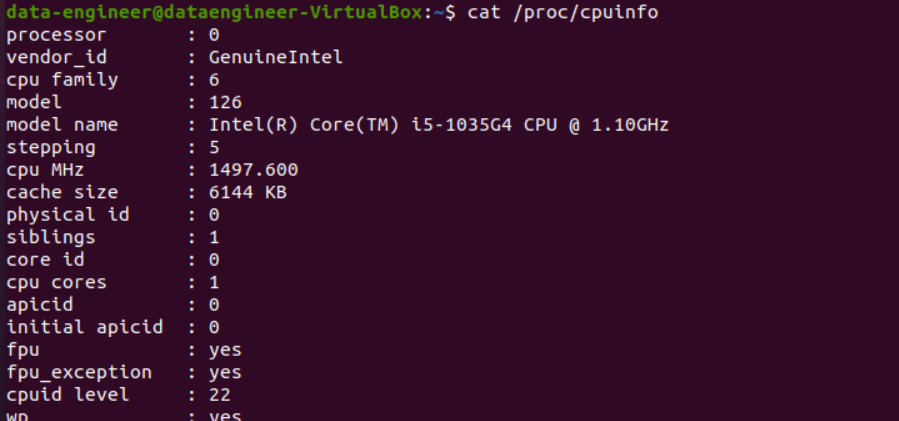
Еще одной полезной утилитой, позволяющей получить информацию о задействовании пространства жесткого диска, является **free**, вот как например можно посмотреть с ее помощью свободную и задействованную память в мегабайтах

****

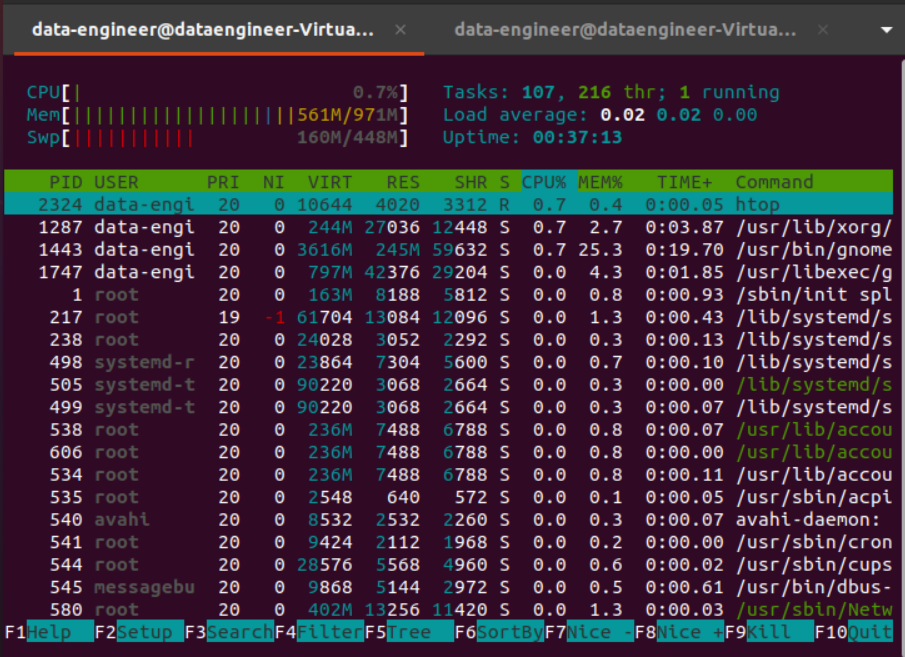
Другим важнейшим ресурсом является вычислительная мощность центрального процессора (CPU, Central Processing Unit) или графического процессора (GPU, Graphical Processing Unit). Получить информацию о CPU можно следующим образом

**cat /proc/cpuinfo**

в результате вы увидите описательную информацию об установленном центральном процессоре и его характеристиках



Получить же информацию об объеме задействования вычислительной процессорной мощности можно с помощью удобной утилиты для мониторинга ресурсов top, либо ее более удобной версии htop. Вот пример графического интерфейса htop

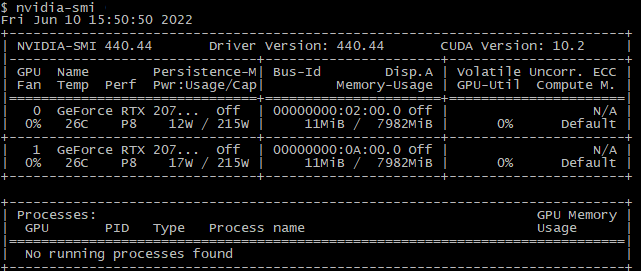


Эта утилита дает нам очень много информации о системе, поэтому давайте подробно разберем содержание этого интерфейса.

|  |  |
| --- | --- |
| CPU | Загрузка центрального процессора, 0.7% от имеющейся производительности |
| Mem | Задействованная оперативная память, 561 Мб из 971 Мб свободных |
| Swp | swap это механизм виртуальной памяти, при котором часть данных из RAM сохраняется на жесткий диск. Позволяет экономить RAM.  Задействованная swap память, 160Мб из 448Мб свободных |
| Tasks | Информация о запущенных задачах (Tasks), являющихся в linux синонимом процесса. Также здесь можно видеть информацию о выполняющихся потоках (thr, threads). Поток отличается от процесса тем, что у процесса собственные, изолированные от других процессов ресурсы, а потоки могут обращаться к одним и тем же ресурсам, например, одной области памяти.  107, 216 thr; 1 running  107 - активных задач (Tasks), которые занимают ресурсы, но могут находится в пассивном режиме, 216 thr - запущенных потоков (Threads), 1 running - одна задача выполняется (в активной фазе) |
| Load average | Средняя загрузка системы за последние 1, 5 и 15 минут  0.02 0.02 0.00 |
| Uptime | Продолжительность работы системы после последнего перезапуска |

Далее идет таблица, в которой перечислены процессы и их параметры: PID, имя, пользователь, время запуска, задействованные процессом ресурсы, приоритет в системе. Эту таблицу удобно использовать при анализе проблем нехватки ресурсов.

Для проектов машинного обучения использования htop может оказаться недостаточным, так как эта утилита не умеет работать с графическими процессорами GPU, в htop нельзя увидеть статистику по задействованию GPU. Для мониторинга и управления GPU необходимо использовать специализированные утилиты, например, для работы с GPU производства NVidia используется утилита nvidia-smi



В консольном выводе утилиты nvidia-smi вы видите характеристики оборудования и информацию о задействовании GPU.

И, в заключении этого раздела, отметим полезную python библиотеку psutil, которую можно использовать прямо в python коде, например в jupyter ноутбуке при проверке имеющихся ресурсов перед началом эксперимента.

**Практическая задача:** Выяснить какая папка имеет наибольший размер в директории /etc.

Подсказка: воспользоваться утилитой ncdu.

1. **Утилиты для работы с сетью**

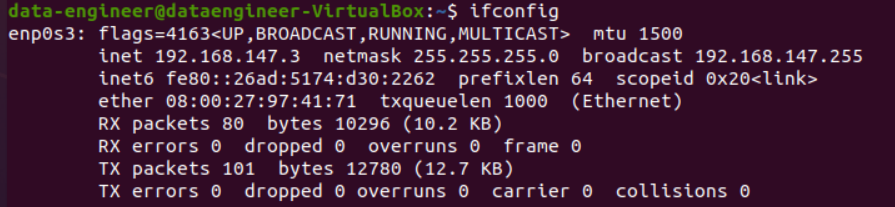
Сети передачи данных часто играют важную роль в проектах машинного обучения, поскольку большинство проектов машинного обучения имеют распределенную инфраструктуру, в которой отдельные узлы связываются друг с другом через сетевые интерфейсы с использованием сетевых протоколов.

Например:

* инженеры данных получают данные по сетевым протоколам от удаленных устройств,
* датасеты после преобразования сохраняются в удаленное сетевое хранилище данных,
* модель машинного обучения обучается на отдельном сервере с мощными GPU, после чего обученная модель сохраняется в git репозитории,
* служебные сервисы MLOps Airflow, MLFlow, Jenkins являются сетевыми и часто работают на отдельных служебных серверах,
* высокопроизводительные базы данных, например PostgreSQL, являются сетевыми и работают на выделенных серверах с возможностью доступа через сеть передачи данных.

Иметь возможность мониторить состояние сетей передачи данных и отдельных сетевых сервисов очень важно для контроля работы системы и диагностики неисправностей. Давайте рассмотрим основные, наиболее часто используемые утилиты.

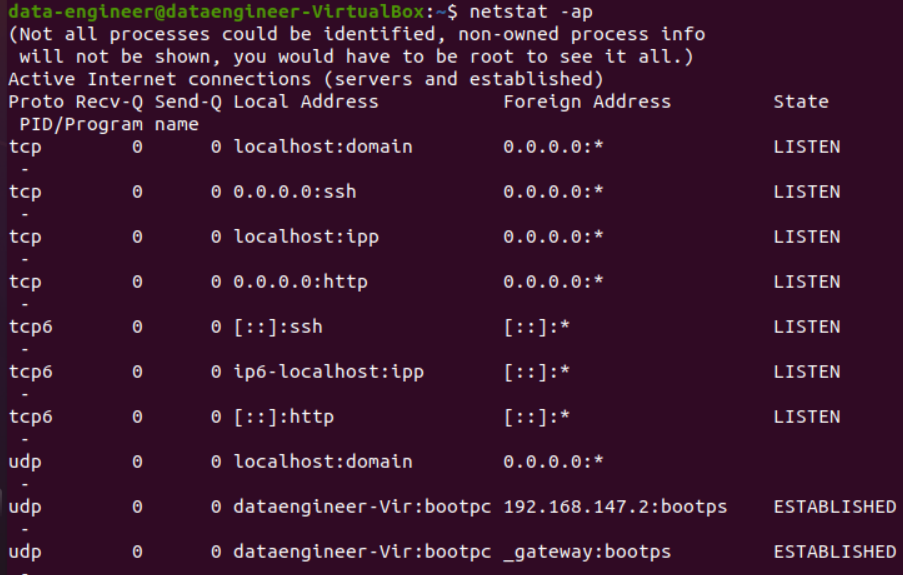
Посмотреть состояние сетевых интерфейсов можно с помощью команды ifconfig



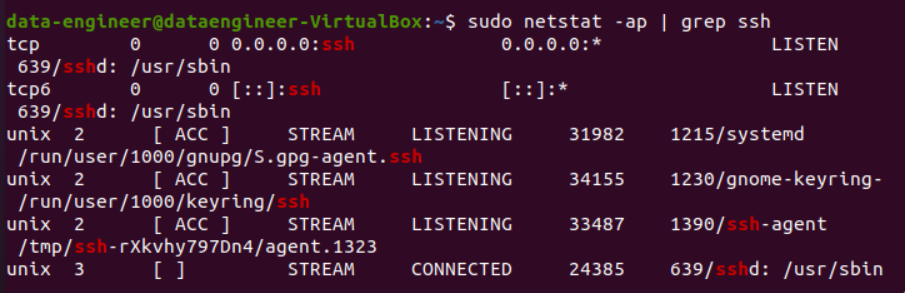
Для доступных сетевых интерфейсов ifconfig показывает

* параметры
  + MAC и IP адреса,
  + маску сети,
  + адрес шлюза,
  + размер максимальной единицы передачи (MTU, Maximum Transfer Unit)
* статистику
  + полученные и принятые пакеты
  + количество ошибок

Более подробную информацию о состоянии сетевых интерфейсов можно получить с помощью команды netstat с различными флагами, более подробную информацию об использовании того или иного флага можно прочитать в man netstat.



В зависимости от используемых флагов с помощью команды netstat можно посмотреть статистику по сетевым сервисам, задействованные порты, типы протоколов, состояния соединений, адреса удаленных подключений и многое другое. Командой netstat удобно пользоваться для диагностики инструментов машинного обучения, использующих сетевые подключения, например Jupyter, Airflow, Jenkins. Например, можно легко выяснить занят ли порт 22 (этот порт зарезервирован за службой ssh в файле /etc/services) следующим образом.



Из этой информации видим, что кроме стандартного демона sshd, который слушает в фоновом режиме входящие соединения на порте 22, в операционной системе есть и другие сетевые процессы ssh.

**Практическая задача:** выяснить в каком состоянии находится порт 8000 (свободен или занят), и, если он занят каким-то процессом, определить что это за процесс и какой у него PID.

Подсказка: воспользоваться утилитой netstat

1. **Утилиты для работы с пользователями и группами**

Вы уже сталкивались в предыдущих модулях с необходимостью создания пользователя с необходимыми привилегиями. Например, для работы с Ansible необходимо было создать пользователей с правами суперпользователя, то есть участника группы sudo, для выполнения команд на удаленных управляемых серверах. Пользователи и группы создаются, модифицируются и удаляются не очень часто, обычно это связано с появлением нового участника в команде проекта, либо с его уходом или изменениями в функциональных обязанностях. Каждый участник команды должен иметь свой аккаунт на рабочих серверах с соответствующими правами и полномочиями. Это позволяет лучше контролировать использование общих системных ресурсов, планировать и контролировать работу, диагностировать и устранять неисправности. Например, вы можете диагностировать непроизводительное расходование оперативной памяти на рабочем сервере, после чего определить, что один из исследователей данных загрузил в pandas.DataFrame большой датасет и забыл закрыть свой jupyter ноутбук перед отпуском. linux позволяет быстро диагностировать причину такой ситуации и принять необходимые меры.

Создать пользователя в linux можно так:

**adduser --ingroup “имя группы” --home ”имя директория” “имя пользователя”**

С помощью флага --ingroup пользователь сразу добавляется в нужную группу, а флаг --home позволяет создать для нового пользователя домашнюю директорию, обычно в папке /home. При этом группа “имя группы” должна существовать. Группы пользователей в linux используются для удобства администрирования большого количества пользователей. Пользователей с похожими задачами объединяют в группы и к группам системный администратор может применять групповые политики, что существенно сокращает время администрирования пользователей. Например, можно создать группы data-researchers, ml-researchers, developers, в которые объединить исследователей данных, исследователей машинного обучения и разработчиков, соответственно. Создать группу можно с помощью следующей команды

**addgroup “имя группы”**

Удалить группу тоже очень просто

**delgroup --only-if-empty “имя группы”**

Посмотреть состав групп можно непосредственно в файле /etc/group, либо с помощью команды

**members**

Посмотреть информацию о пользователе

**id username**

Добавить пользователя в группу (-a обязательно add, иначе при выполнении команды будет удалена информация о текущих группах пользователя)

**usermod -a -G docker “имя пользователя”**

В этом примере пользователь добавляется в группу docker.

Удалить пользователя из группы

**gpasswd –d user group**

либо

**userdel user group**

В процессе работы может возникнуть необходимость изменений в пользовательских параметрах. Например, пользователь может перейти в другой отдел или ему понадобятся расширенные права доступа. Такие изменения делаются с помощью утилиты usermod, например вот так можно изменить рабочий каталог пользователя:

**usermod –d “новый каталог” “имя пользователя”**

Поменять пользователя и группу для файла

**сhown ”имя пользователя”:”имя группы” “имя файла”**

Иногда системному администратору необходимо выполнить определенные действия от имени конкретного пользователя. Поменять пользователя можно следующей командой

**su “имя пользователя”**

или

**sudo su “имя пользователя”** (поменять без пароля, воспользовавшись правами суперпользователя).

При расследовании причин неисправностей бывает полезно знать последние действия пользователей в системе, например, с помощью команды

**last -a**

Дата последнего лога каждого пользователя выводится с помощью команды

**lastlog**

Увидеть кто из пользователей в настоящее время находится в системе можно с помощью команды

**who**

Перечисленные утилиты позволяют управлять пользователями и контролировать их работу.

**Практическая задача:** установить в систему docker, добавить пользователя и добавить этого пользователя в группу docker, чтобы он мог выполнять команды docker без sudo

Подсказка: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-ru

1. **Утилиты для работы с процессами.**

Кроме пользователей в linux важную роль играют процессы, выполняющие задачи. Например, для работы JupyterHub запускается процесс, обеспечивающий работу пользовательского интерфейса, авторизацию, выполнение скриптов. Каждый процесс использует общие системные ресурсы. Иметь инструмент для обнаружения “зависших” процессов, либо несанкционированно запущенных, очень удобно.

Посмотреть процессы, работающие в системе, можно следующей командой:

**ps**

или, для более подробного вывода

**ps –aux**

Например, вот так можно найти все процессы с названием jupyter

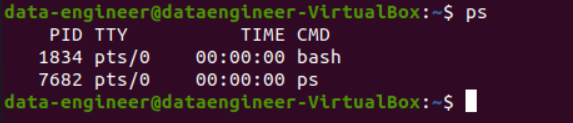
**ps -aux | grep jupyter**

Вертикальная черта “|” передает результат выполнения команды “ps -aux” на вход команды “grep jupyter”. Такая конструкция в linux называется конвейер. При отладке пайплайнов машинного обучения, состоящих из отдельных скриптов \*.py вы можете использовать такой конвейер, например в виде

**python get\_data.py | python data\_proc.py | train\_model.py**

Но такой вид скорее подходит для упрощенных экспериментов на самых начальных этапах по исследованию данных и созданию моделей.

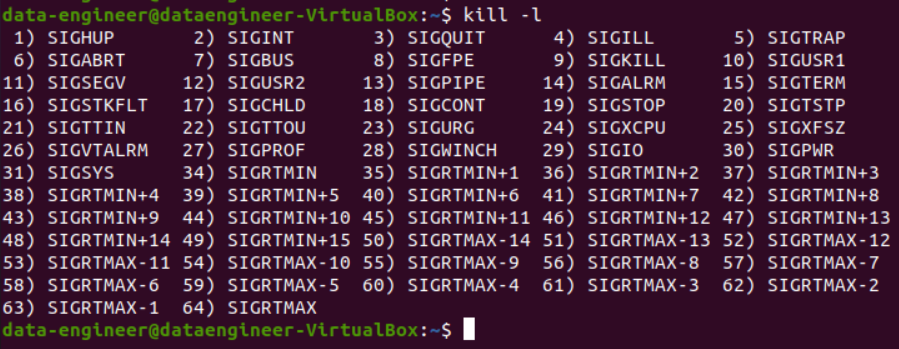
С любым процессом в linux связано несколько важных параметров, которые вы видите при результате выполнения команды ps



Например, PID это идентификатор процесса (Process IDentificator), уникальный номер процесса в системе. Если вам понадобится срочно остановить процесс, то вы можете это сделать с помощью команды

**kill -9 “PID процесса”**

Несмотря на устрашающее название команда kill используется всего лишь для отправки различных сигналов процессам, сигналы это сообщения, которыми процессы обмениваются между собой, сигнал “-9” означает “SIGKILL”, то есть “остановить процесс”, полный список возможных сигналов:

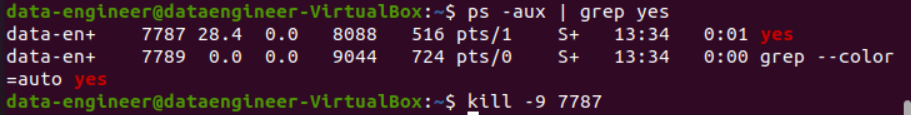


Давайте разберем практическую задачу: в одном терминале запустим утилиту yes, которая будет постоянно выводить символ “y” на экран. Кстати, эти используется для автоматизации установки программного обеспечения, в которой требуется ввести символ “y” (yes). А в другом терминальном окне мы найдем этот процесс, узнаем его уникальный PID и остановим с помощью команды kill.

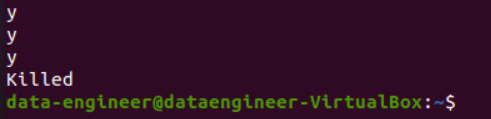
Запустим yes и в терминал начнут выводится символы “y” (либо те, которые мы укажем)



В другом окне проделаем следующее



После этого процесс выполнения утилиты “yes” будет остановлен и в первом терминальном окне мы увидим следующее



Рекомендуем пользоваться утилитой kill очень осторожно, чтобы случайно не остановить какой-то нужный процесс по ошибке.

**Практическая задача:** узнать потребление ресурсов (оперативная память и процессор) процессов, участвующих в обучении модели ML.

Подсказка: воспользоваться ps и htop.

1. **Штатные текстовые редакторы**

Часто в процессе эксплуатации возникает необходимость быстро вручную исправить конфигурационные файл или посмотреть содержимое файла. Для этих целей удобно использовать типовые текстовые редакторы linux. Например vi/vim, nano, emacs, nano и другие. Хороший обзор редакторов и некоторых правил работы с ними можно найти здесь: <https://losst.ru/luchshie-tekstovye-redaktory-linux>. Многие редакторы имеют не очень понятную логику использования, особенно для пользователей Windows, привыкших работать с MS Word. Однако после ознакомления с основными приемами работы в текстовых редакторах linux пользователи понимают их удобство и преимущество для решения эксплуатационных задач. Рекомендуем вам ознакомиться с одним из этих редакторов самостоятельно, чтобы при выполнении практических задач вы могли быстро вносить исправления в конфигурационные файлы или искать в них нужную информацию.

1. **Управление окружением**

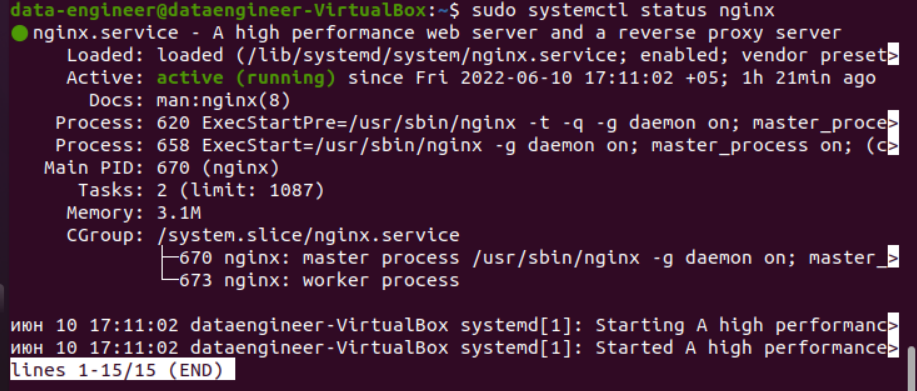
Информация о глобальном окружении операционной системы находится в файле /etc/environment, который можно посмотреть с помощью текстового редактора, например vi

**vi /etc/environment**

В /etc/environment сохраняются значения переменных окружения, в частности PATH.

1. **Управление работой сервисов**

С помощью утилиты systemctl можно управлять работой сервисов, например, запускать, останавливать, перезапускать, узнавать текущий статус. Выполнение этой команды требует наличие прав суперпользователя (sudo). Вот как можно узнать текущий статус сервера nginx.



Из информации видно, что web-сервер nginx успешно запущен, находится в состоянии active (running). Этот сервис можно остановить с помощью команды

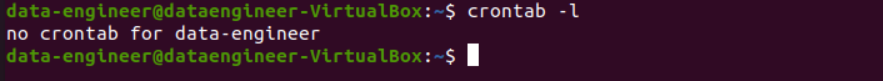
**sudo systemctl stop nginx**

или перезапустить с помощью команды

**sudo systemctl restart nginx**

Запуск задач по расписанию осуществляется с помощью crontab.

Например, увидеть актуальное расписание для задач можно с помощью команды crontab -l



Чтобы перейти к созданию расписания для выполнения задачи надо использовать команду crontab -e. Для создания расписания используется специфический crontab синтаксис

# строки-комментарии

#

# задача, выполняемая ежеминутно

\* \* \* \* \* “команда”

# задача, выполняемая на 10-й минуте каждого часа

10 \* \* \* \* “команда”

# задача, выполняемая на 10-й минуте второго часа каждого дня и использующая перенаправление стандартного потока вывода

10 2 \* \* \* “команда” > /tmp/logfile.log

Вот полезный ресурс для того, чтобы осуществлять преобразование форматов в crontab https://crontab.guru/

1. **Средства для удаленной работы**

* rsync синхронизация файлов и папок на разных хостах
* wget скачать файл
* nmap проверить порты
* telnet незащищенное консольное подключение
* ftp передача файлов
* ssh защищенное консольное подключение
* curl отправить информацию на удаленный сервер

**Практическая задача:** скачать удаленно файл из Интернет.

Подсказка: воспользоваться wget

Конечно, в linux существует еще масса полезных инструментов, как консольных, так и графических. Как и для любых других инструментов, уверенность в использовании приходит с опытом, поэтому рекомендуем вам практиковаться в использовании команд linux для решения типовых задач.

*Тест*

1. Какие команды используются для проверки доступного места на жестком диске? (0.25)
   1. **df**
   2. netstat
   3. grep
   4. **free**
2. Какая команда позволяет определить процесс, занимающий порт 8000? (0.25)
   1. sudo get port 8000
   2. **sudo netstat -ap | grep 8000**
   3. sudo what port 8000
   4. sudo ifconfig port 8000
3. Какая утилита позволяет выполнять команды от имени суперпользователя? (0.25)
   1. su
   2. superuser
   3. **sudo**
   4. do
4. Какая утилита позволяет администрировать GPU? (0.25)
   1. gpu-util
   2. **nvidia-smi**
   3. hw-util
   4. gpu-admin

*Итоги/выводы*

Большинство проектов машинного обучения используют в качестве операционной системы linux. Поэтому инженеры DevOps/MLOps должны уметь пользоваться необходимыми утилитами linux для эффективного мониторинга и управления ресурсами проекта. В этом юните вы познакомились с утилитами для системного администрирования операционной системы linux, которые вам обязательно пригодятся при создании и настройке инфраструктуры, мониторинге и поиске неисправностей.

Использование этих инструментов предполагает уверенное владение консолью для выполнения команд, что бывает неудобно для визуализации сложных процессов. Следующие юниты посвящены инструментом с более удобным графическим интерфейсом, что позволяет осуществлять мониторинг и управление программными системами без выполнения команд в консоли.

# Модуль 8. Юнит 2. Утилита Tensorboard для мониторинга.

*Введение:* В предыдущих модулях вы уже узнали о различных инструментах автоматизации процесса машинного обучения, которые имеют средства визуального контроля выполнения операций, например Airflow, Jenkins. Имеющиеся в Airflow и Jenkins штатные средства визуализации не позволяют осуществлять подробный мониторинг процессов в системе. Например, в процессе обучения модели машинного обучения встроенные средства визуализации Jenkins или Airflow сигнализируют только о том, что тот или иной процесс завершился аварийно, а при необходимости узнать детали требуется изучение сообщений в консольном выводе приложений. Для оперативного мониторинга важных процессов в проектах машинного обучения используется специальный подход, когда выполняющийся скрипт или программа пишут в специальный log файл информацию о параметрах своей работы в реальном времени, а специальные средства визуализации читают информацию из этого log файла и отображают прочитанные параметры в графическом виде. Такой подход позволяет своевременно увидеть негативные тенденции во внутренних выполняющихся процессах. Кроме рассмотренных ранее общесистемных параметров (оперативная память, процессор, жесткий диск, сетевые интерфейсы), к важным параметрам, которые нужно наблюдать и контролировать, относятся:

* метрики качества модели машинного обучения в процессе обучения и инференса,
* количество ошибок в поступающих данных,
* процент выполнения задачи.

В этом юните мы изучим инструмент Tensorboard, а в следующем юните вы узнаете про его более универсальный и широко распространенный аналог, Grafana. Встроенный во фреймворк Tensorflow инструмент Tensorboard позволяет осуществлять мониторинг процесса обучения модели в Tensorflow, например, контролировать метрики, предупреждать переобучение. Благодаря наглядности и интерактивности Tensorboard является удобным инструментом контроля процесса машинного обучения и эксплуатации модели.

*Содержание юнита:*

Одна из важных задач в процессе машинного обучения это контроль переобучения модели. С этой целью обычно анализируют в процессе обучения качество работы модели на тренировочных данных, на которых модель учится, и тестовых данных, которые модель ранее не видела. Если в процессе обучения при росте качества работы модели на тренировочных данных стало ухудшаться качество работы на тестовых данных, то это означает, что модель переобучилась и дальнейшее обучение не имеет смысла.

Этот процесс можно контролировать штатными средствами python, например с помощью графиков, построенных в библиотеках для визуализации: matplotlib, seaborn или plotly. При этом сначала выполняется обучение модели, анализируется качество работы модели на обучающей и тестовой выборках на каждой эпохе, а после обучения модели эти результаты для каждой эпохи представляются на графиках. Вот, например, как можно получить информацию о ходе обучения модели в Tensorflow. Сначала результаты обучения на каждой из эпох сохраняются в специальных словарях history.history[‘loss’] и history.history[‘val\_loss’]

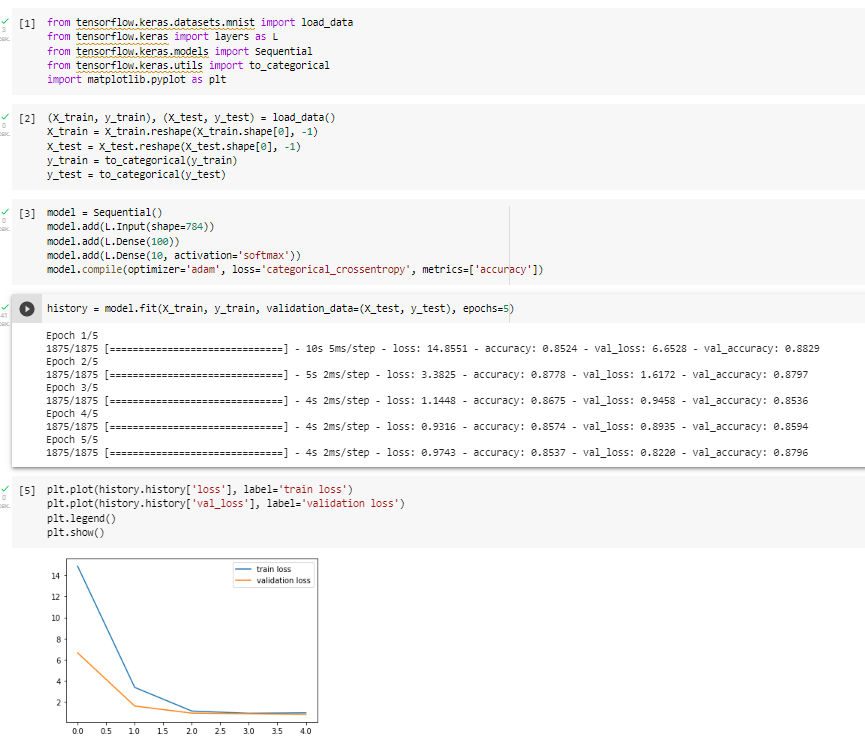
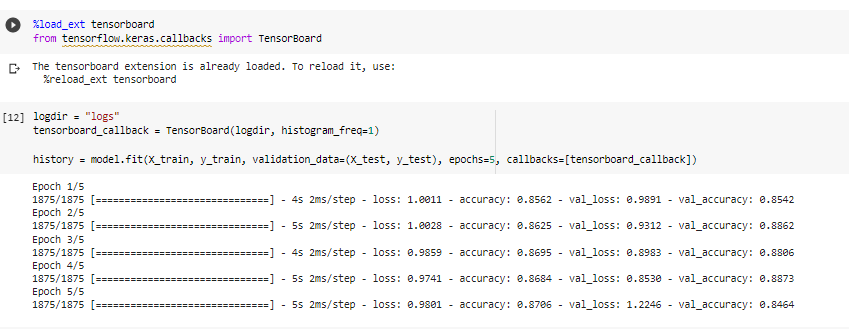


Рисунок “Пример обучения модели и визуализации потерь на разных эпохах”

То есть мы визуализировали информацию о динамике изменения loss (потери на тренировочной выборке) и val\_loss (потери на валидационной выборке) штатными средствами matplotlib. И если в процессе обучения имеет место переобучение модели, то есть ситуация, при которой качество модели на тренировочных данных растет, а на валидационных постоянно снижается, это можно будет легко увидеть на графике. Такой подход позволяет проанализировать только уже завершенный процесс обучения. Есть инструменты, позволяющие мониторить процесс обучения в реальном времени. Один из таких инструментов TensorBoard, который является штатным средством TensorFlow. Использование TensorBoard предполагает запись в log-файлы на каждом этапе выполнения обучения, TensorBoard считывает информацию из этих log-файлов и показывает данные в удобном графическом формате.

Давайте сначала рассмотрим пример использования TensorBoard в среде google colab (https://colab.research.google.com), который позволяет обращаться к Tensorboard с использованием “магических команд” Jupyter ноутбука в google colab, %load\_ext tensorboard и %tensorboard. Вне google colab эти магические команды не работают, для этих ситуаций использование tensorboard описано далее в юните. При использовании tensorboard в google colab сначала надо загрузить tensorboard с использованием “магической” команды %load\_ext tensorboard. Далее создается и обучается модель и при обучении идет сохранение информации о параметрах модели с использованием механизма функций “обратного вызова” (“функций-колбаков”, callbacks). Функции “обратного вызова” срабатывают при наступлении определенного события, в нашем случае они выполняются при завершении каждой эпохи обучения.



В процессе обучения можно вызвать TensorBoard с помощью магической команды %tensorboard и увидеть состояние параметров модели в режиме реального времени

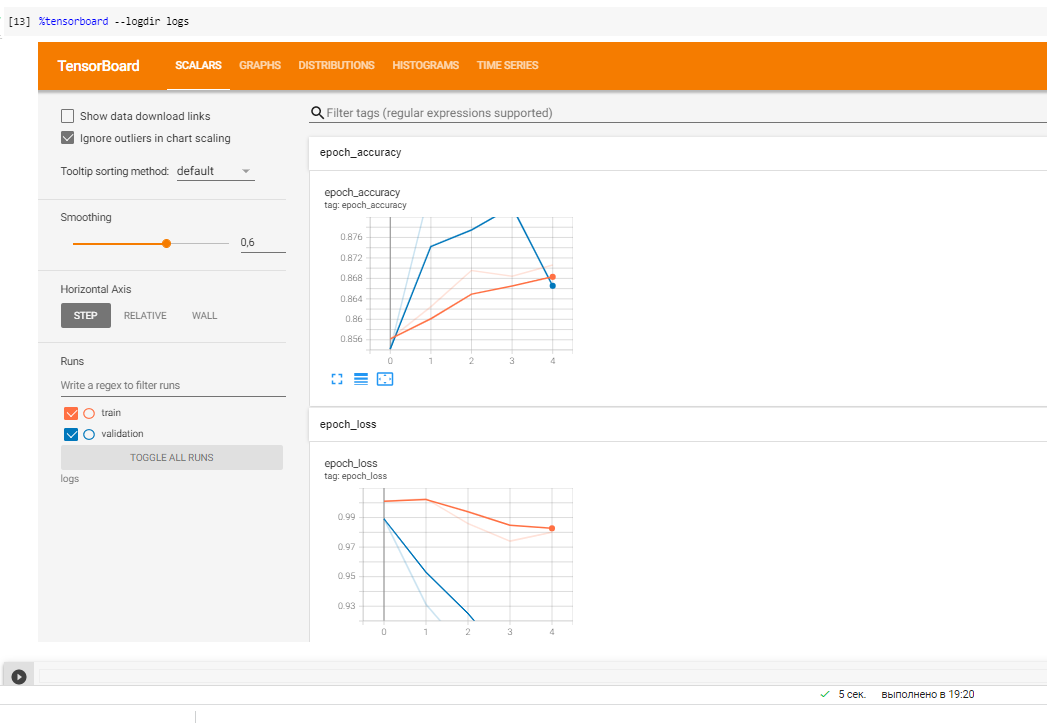
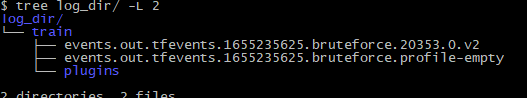
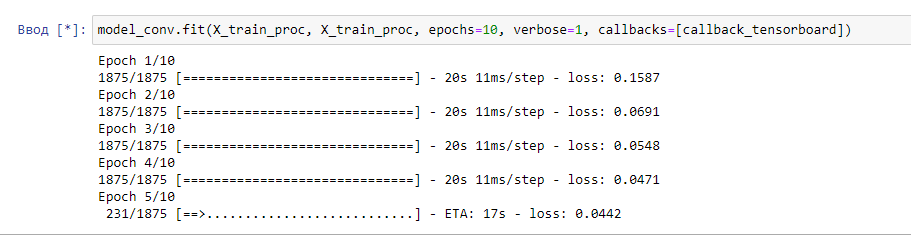


Рисунок “Графический интерфейс tensorboard”

При этом tensorboard является сетевым приложением и к нему можно подключиться с удаленного компьютера или системы, чтобы получить актуальную информацию о ходе процесса обучения.



Запустим продолжительное обучение модели с указанием функции обратного вызова, записывающей информацию в log файл.



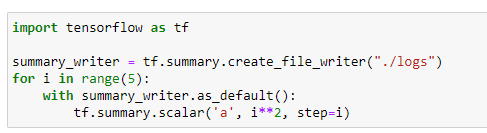
Одновременно запустим сервис tensorboard на том же сервере, где сохраняется log файл

**tensorboard --logdir=”имя лог файла”**



Видим информацию о состоянии loss в графическом интерфейсе.

Конечно, с помощью функций обратного вызова можно выводить только информацию, касающуюся обучения, а для конкретных производственных задач может потребоваться большая гибкость в возможностях вывода. Например, мы хотели бы отслеживать в Tensorboard использование оперативной памяти или значение какой-то внутренней переменной программы. В Tensorboard поддерживается такая операция записи с использованием специального объекта writer, создаваемого с помощью функции tensorflow.summary.file\_writer().



В приведенном коде происходит запись квадрата величины *i* в лог файл. При этом в графическом интерфейсе Tensorboard мы увидим следующий график изменения этой величины. Используя этот механизм можно записывать в log файл для последующего отображения в графическом интерфейсе Tensorboard любую полезную информацию, которую необходимо контролировать.

*Практическое задание:*

Создайте свою произвольную модель машинного обучения в Tensorflow и визуализируйте процесс ее обучения с TensorBoard.

Результат выполнения задания: jupyter ноутбук, который можно выполнить в google colab.

*Итоги/выводы*

В связи с тем, что процесс обучения модели машинного обучения является очень продолжительным по времени, необходимы инструменты для мониторинга этого процесса в реальном времени, чтобы своевременно принимать меры по устранению возможных неполадок. Это позволит сэкономить значительное количество времени, поскольку аппаратное обеспечение не будет загружено бесполезными вычислениями. В этом юните вами был изучен инструмент Tensorboard для визуализации процесса машинного обучения в реальном времени. В качестве практической задачи рассмотрено применение TensorBoard в среде google colab и утилита tensorboard в linux. В следующем юните вы узнаете о более распространенном инструменте Grafana, который применяется не только в проектах машинного обучения.

# Модуль 8. Юнит 3. Визуализация и контроль с использованием Grafana.

*Введение:*

Любая сложная технологическая система содержит множество параметров, влияющих на ее работу. Мониторинг этих параметров позволяет своевременно заметить опасные тенденции в работе системы и принять меры по их устранению. Как известно, проблему гораздо проще предотвратить, чем устранять последствия аварии. Анализировать большое количество параметров человеку очень непросто. Даже самый опытный и внимательный инженер имеет предел своей производительности, связанный с физическими ограничениями. Чрезмерная нагрузка на специалиста, занимающегося обслуживанием сложных технологических систем может привести к ошибкам, связанным с “человеческим фактором”, поскольку человек не может работать двадцать четыре часа в сутки продолжительное время и не может постоянно анализировать данные, поступающие от сотен различных систем. Обслуживание сложных технологических систем невозможно без специальных инструментов мониторинга и автоматизированного контроля, являющихся незаменимыми помощниками в процессе эксплуатации. Такие инструменты умеют собирать данные из различных источников, визуализировать эти данные, уведомлять специалистов при возникновении какого-то нарушения и многое другое. В этом юните вы узнаете об одном из таких инструментов, фреймворке Grafana. Полученные сведения и практические навыки вы сможете применить для добавления качественной визуализации процессов мониторинга и контроля с использованием Grafana в любой из этапов проекта машинного обучения, начиная от сбора и анализа данных и заканчивая процессом эксплуатации модели. Не умаляя важности всех этапов проекта машинного обучения следует отметить, что без обеспечения качественной процедуры эксплуатации модели в производственной среде все другие усилия и достижения команды проекта могут оказаться бесполезными, так как для заказчика польза системы будет сомнительной из-за плохих результатов в эксплуатации. Поэтому в изложении возможностей Grafana в этом юните сделан акцент именно на эксплуатационных задачах.

В результате изучения этого юнита вы узнаете:

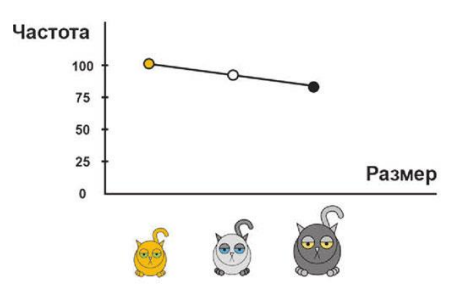
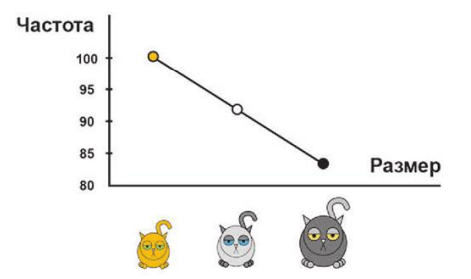
* историю создания и развития фреймворка Grafana, его основные функциональные возможности,
* процедуру установки Grafana,
* варианты применения Grafana в проекте машинного обучения.

*Содержание юнита:*

В проектах машинного обучения часто встречаются участники, не владеющие навыками работы с данными, не умеющие обучать модели машинного обучения, не разбирающиеся в математике и статистике. Большинство людей и, в том числе, наших коллег, не обладают теми знаниями, которыми обладают исследователи данных или инженеры машинного обучения. Тем не менее такие специалисты могут играть очень важную для проекта роль: заниматься поддержанием работоспособности проекта на производственной площадке, работать с результатами работы модели или даже принимать решение о финансировании такого проекта машинного обучения. Во всех перечисленных случаях для инженера эксплуатации (обслуживающего систему), операциониста (запрашивающего у модели машинного обучения результаты прогноза) или для финансового директора характеристики, особенности и результаты работы проекта машинного обучения должны быть переведены на понятный им язык и, чаще всего, это содержательные и интерпретируемые графики. Чем более понятным и содержательным будет график, тем правильнее будет принято решение, основанное на данных, представленных на этом графике, даже если специалист, принявший это решение, не обладает глубоким пониманием нюансов работы модели.

Визуализация является важнейшей частью современных программных систем, которая относится к так называемому UX/UI (User eXpirience / User Interface, пользовательский опыт / пользовательский интерфейс) и означает специальные подходы к созданию удобного, непротиворечивого, эргономичного, эффективного пользовательского интерфейса для работы с приложениями. UX/UI посвящено много тематической литературы, но в данном юните мы будем рассматривать важность качественного пользовательского интерфейса только в части использования эффективных графиков и дашбордов, содержащих визуальную информацию, необходимую для качественной эксплуатации программных систем и, в том числе, проектов машинного обучения.

То, что правильная визуализация влияет на восприятие, можно увидеть в интересном и показательном примере из книги Владимира Соловьева “Статистика и котики”, в которой в игровой форме излагаются сложные понятия и инструменты статистического анализа. Благодаря простоте и оригинальности изложения эта книга стала бестселлером. Ниже приведены рисунки из этой книги. На левом рисунке график, из которого следует, что при изменении размера происходит существенное изменение частоты появления особи такого размера. А вот на правом графике это изменение уже не такое радикальное, при увеличении размера график частоты меняется незначительно. Курьезность ситуации заключается в том, что на левом и правом графиках описан один и тот же процесс! Приглядитесь внимательно и вы поймете насколько может визуализация поменять представление об описываемом процессе или демонстрируемых результатах.



Таким образом, качественная визуализация позволяет лучше понять процесс, который эта визуализация описывает.

Современные средства, используемые в процессе эксплуатации сложных систем, используют специальные инструменты и подходы, которые позволяют эффективно решать задачи эксплуатации:

* интерактивные дашборды, позволяющие визуализировать данные с изменением параметров, увидеть изменение процесса в динамике, при меняющихся условиях, задать фильтры или дополнительные условия для того, чтобы получить более информативную и понятную визуализацию,
* работа с многомерными данными и временными рядами, возможности для анализа их взаимосвязи,
* автоматизированный мониторинг значений параметров в заданных диапазонах, при превышении порога срабатывания (так называемой уставки), система может уведомить персонал эксплуатации о превышении,
* различные системы аналитики
  + описательная (дескриптивная) аналитика описывает уже произошедшие события
  + предиктивная аналитика позволяет предсказывать события
  + предписывающая аналитика дает рекомендации для принятия оптимальных действий
* типовые интерфейсы для автоматизации и взаимодействия с другими системами (SQL, JSON)
* подсистему авторизации пользователей, позволяющую предоставлять информацию только тем пользователям, которые авторизованы для получения этой информации.

В настоящее время создано большое количество библиотек, фреймворков, open-source и коммерческих продуктов, которые решают перечисленные выше задачи. Grafana, один из самых популярных open-source проектов, предназначен для визуализации данных, алертов – независимо от того где находятся данные. Мы рассмотрим этот инструмент в данном юните.

Вся необходимая для работы информация о Grafana находится на официальном сайте этого проекта [https://grafana.com](https://grafana.com/). Здесь можно найти все необходимые инструкции по работе, примеры, программное обеспечение. Grafana был создан в 2014 году, с тех пор этот инструмент приобрел огромную популярность. Grafana это платформа с открытым программным кодом (open-source), которую используют для визуализации, мониторинга и анализа данных. Grafana может взаимодействовать с множеством систем, которые являются для нее “источниками” данных для визуализации. В соответствии с поддерживаемой архитектурой «подключаемых» источников данных, Grafana имеет модули интеграции для большинства типовых инструментов работы с данными:

* реляционные базы данных: MySQL, PostgreSQL и др.
* нереляционные базы данных: Graphite, Prometeus, Elasticsearch, Open TSDB, InfluxDB и др.
* служебные системы мониторинга облачных сервисов: Google StachDriver, MS Azure, Amazon CloudWatch и др.

Для каждого источника, с которым взаимодействует Grafana, поддерживается синтаксис этого источника, реализуемый в специальном редакторе запросов.

Grafana имеет возможность быстрого создания учебного макета, на котором любой может ознакомиться с возможностями Grafana, поэкспериментировать с настройками и выполнением типовых действий по администрированию и использованию. Это учебное окружение для изучения grafana создается очень просто с использованием следующих действий:

* изучить описание по адресу <https://grafana.com/tutorials/grafana-fundamentals/?pg=docs>, установить необходимое, описанное в этой документации, программное обеспечение, например, docker, docker-compose и git,
* клонировать к себе официальный репозиторий проекта с помощью команды

**git clone https://github.com/grafana/tutorial-environment.git**

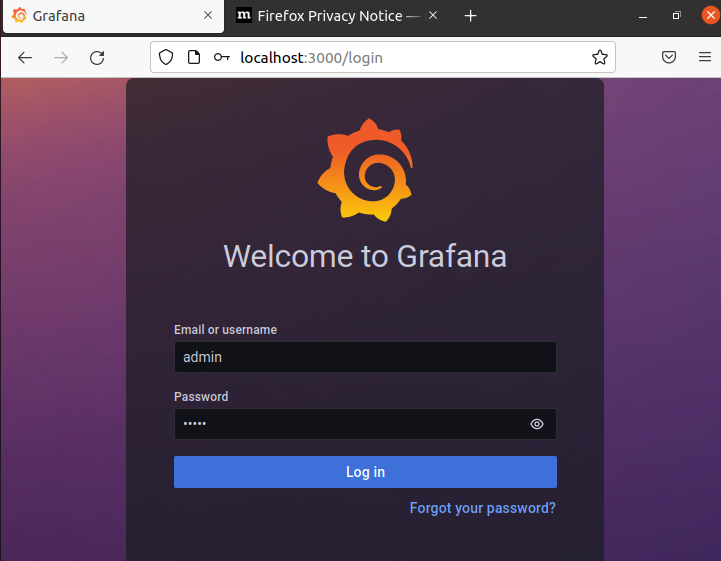
* перейти в рабочую папку с помощью команды

**cd tutorial-environment/**

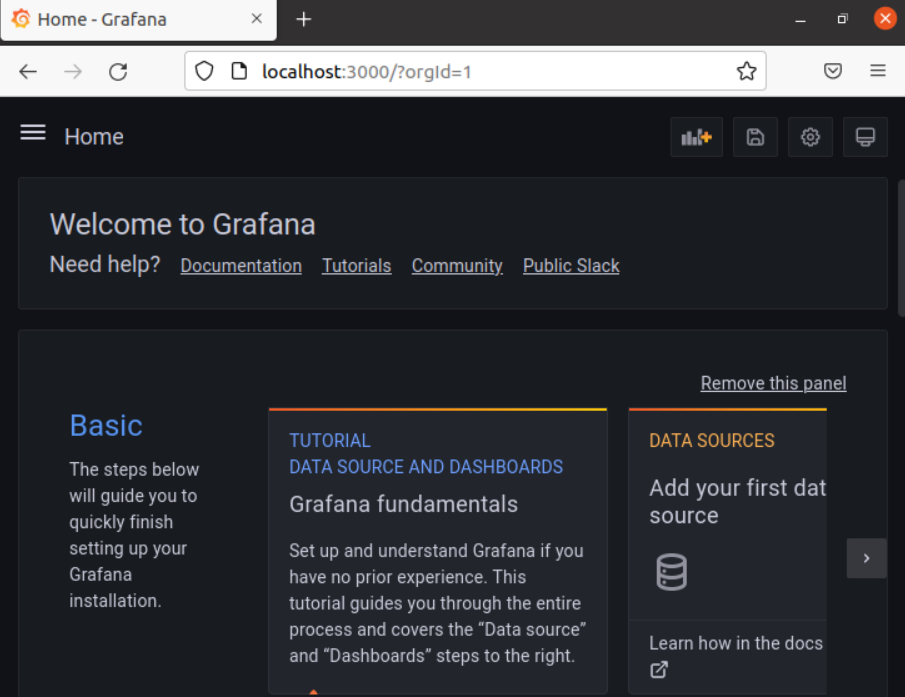
* запустить контейнеры

**docker-compose up**

* далее можно открыть с помощью браузера приложение на локальном хосте по адресу: hostname:3000 (по умолчанию имя пользователя admin, пароль admin, конечно же после логирования в системе рекомендуется эти параметры изменить)
* теперь можно использовать систему, например добавлять источники данных во вкладке Configuration -> Add Data->Prometeus.



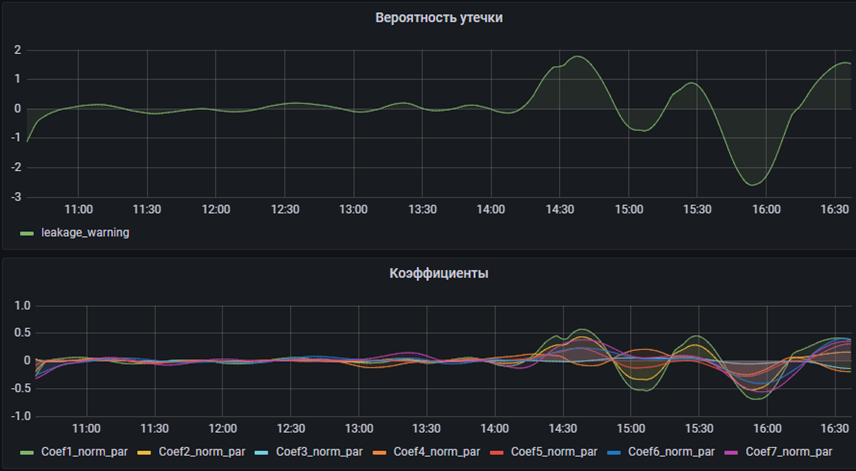
При первом входе в систему вам будет предложено изменить реквизиты входа, рекомендуется это сделать, так как реквизиты admin:admin являются небезопасными, их использование делает систему уязвимой. После этого вы попадаете в основной интерфейс Grafana, который имеет следующий вид



Для дальнейшей работы необходимо понимать основные определения, которые используются в визуализациях Grafana:

* **Панель** Grafana это базовый элемент визуализации выбранных данных. Поддерживается работа с различными видами панелей: с графиками, статусами, таблицами, [тепловыми картами](http://docs.grafana.org/features/panels/heatmap/) и произвольным текстом, а также интеграцию с полезными плагинами (например, карта мира или часы) и приложениями, которые также можно визуализировать. Можно настроить стиль и формат каждой панели; все панели можно переносить на другое место, перестраивать и изменять их размер.
* **Дашборд** это набор отдельных панелей, сопровождающийся набором переменных, которые можно менять. При изменении переменных меняется и содержание дашборда. Все дашборды можно настраивать с учетом конкретных технических требований проекта. Благодаря поддержке большого сообщества в Grafana есть большой выбор готовых дашбордов для разных типов данных и источников.
* В дашбордах можно использовать **аннотации** для отображения определенных событий на разных панелях. Аннотации добавляются настраиваемыми запросами, на графике аннотация отображается вертикальной красной линией. При наведении курсора на аннотацию можно получить описание события и теги. Благодаря этому можно легко сопоставить время, конкретное событие и его последствия в приложении и исследовать поведение системы.

Вот примеры дашбордов Grafana



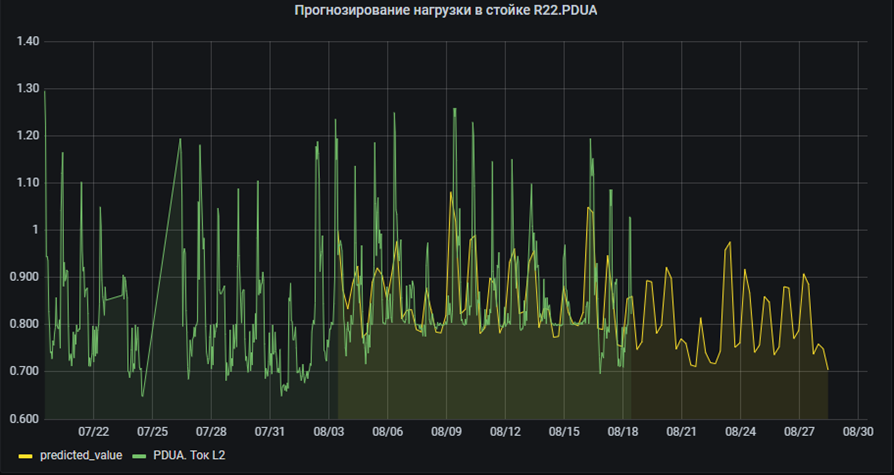
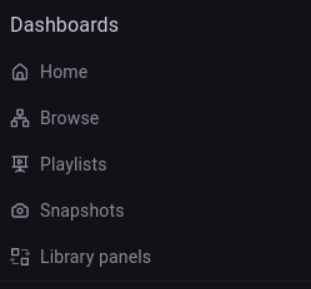
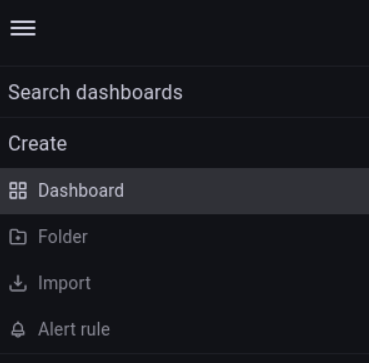
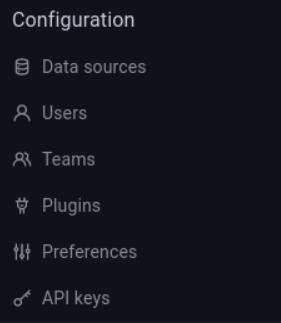


Рисунок “Примеры дашбордов (интерактивных панелей) в Grafana”

Для начала работы наиболее полезными являются разделы “Подключиться к источнику данных”, “Создать дашборд”, “Посмотреть готовые дашборды”



Давайте рассмотрим пример подключения Grafana к базе данных PostgreSQL, расположенном на другом сервере, и отображения на дашборде данных из этой базы данных. В операционной системе Ubuntu база данных PostgreSQL устанавливается очень просто, с использованием команд

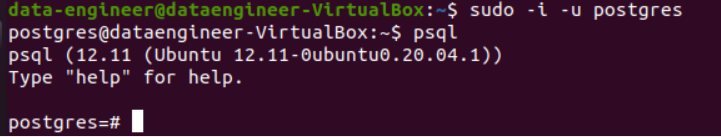
**sudo apt update**

**sudo apt install postgresql postgresql-contrib**

После этого на вашем сервере будет установлена база данных PostgreSQL и служебные утилиты для этой базы. База данных postgresql поддерживает различные механизмы авторизации пользователей, по умолчанию используется концепция пользователей и ролей linux, при установке postgresql создается пользователь postgres, к которому можно перейти с использованием команды

**sudo -i -u postgres**

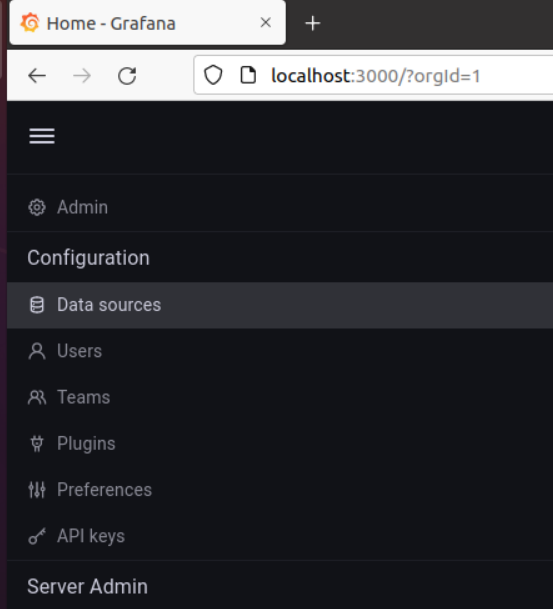
после этого можно вызвать утилиту **psql** и перейти к командной строке для взаимодействия с postgresql.



Теперь, после подключения в базе данных postgresql пользователем postgres мы можем создавать базы данных, таблицы и вносить в них данные с помощью SQL запросов. Команды, с помощью которых можно выполнить эти действия, приведены ниже. Рекомендуем вам самостоятельно изучить взаимодействие с базой данных postgresql и SQL запросы, в частности, можно воспользоваться информацией по ссылкам в конце модуля.



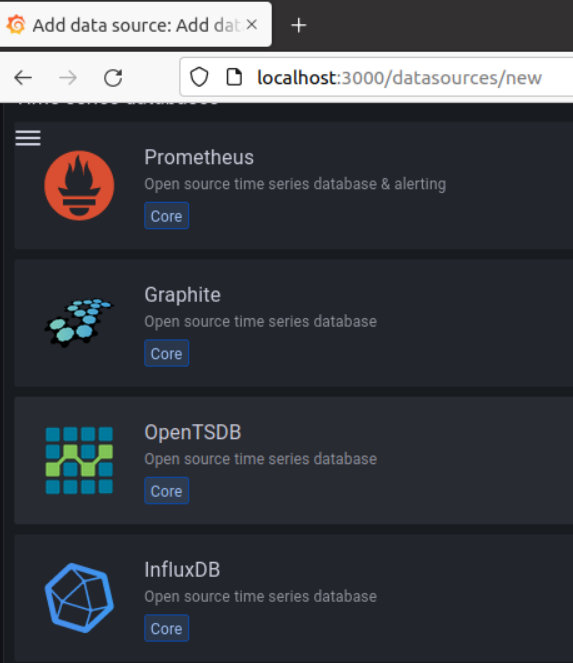
Теперь у вас в базе данных находятся два измерения для accuracy (поле acc) и validation accuracy (поле val\_acc). Давайте отобразим их в grafana. Для этого сначала надо настроить в grafana источник данных (Data Source)



В появившемся диалоговом окне нажимаем клавишу “Add data source” (добавить источник данных).



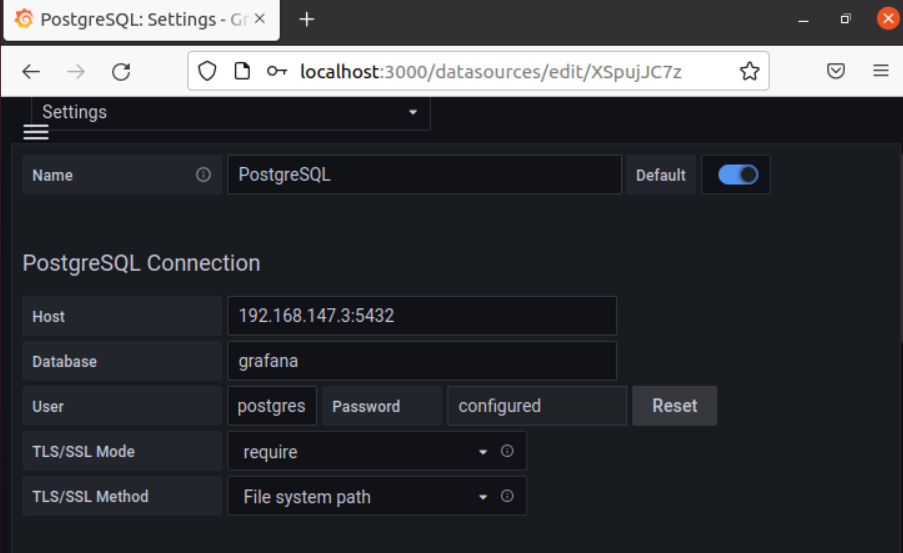
Вы увидите большое количество различных вариантов источников данных для grafana



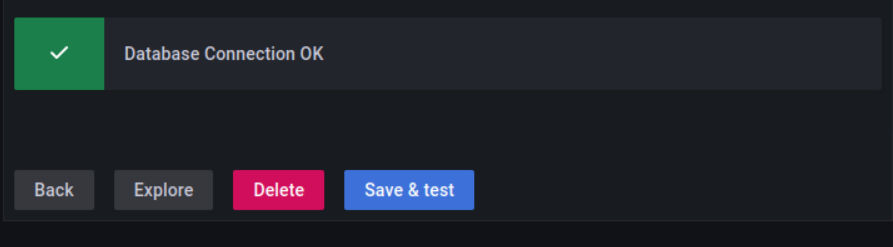
Поскольку мы предполагаем работать с postgresql, то надо выбрать этот источник



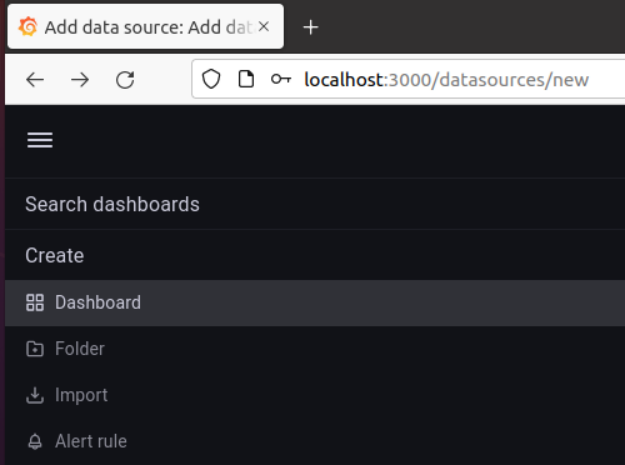
Необходимо вписать правильные реквизиты доступа к удаленному источнику данных для Grafana. В случае с postgresql вам необходимо внести следующие данные:



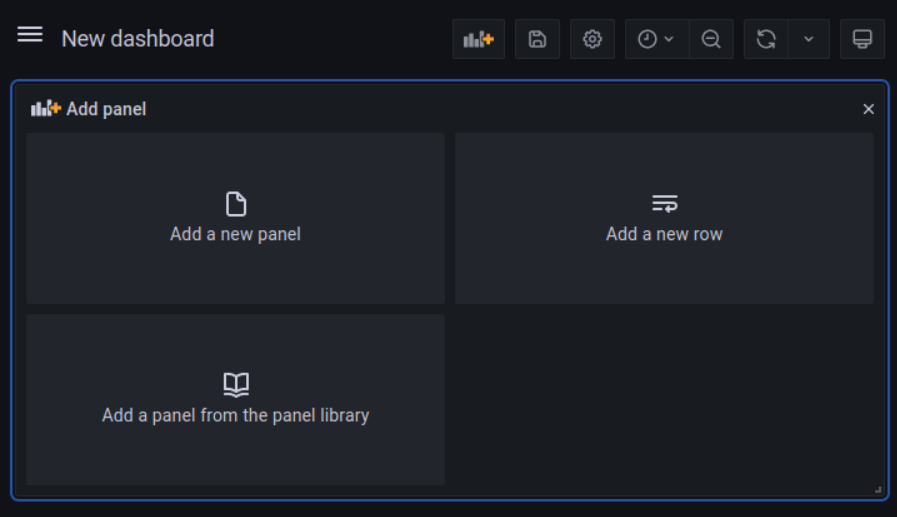
Конечно же вы должны использовать тот IP адрес, на котором установили postgresql, а вот номер порта для подключения к postgresql как правило стандартный, зарезервированный за postgresql, он имеет значение 5432. Также надо ввести правильное имя пользователя и его пароль и название базы данных, из которой вы хотите читать информацию. После этого можно проверить соединение, нажав клавишу “Save & test” (сохранить и протестировать), и если все в порядке, то вы увидите сообщение “Database Connection OK”.



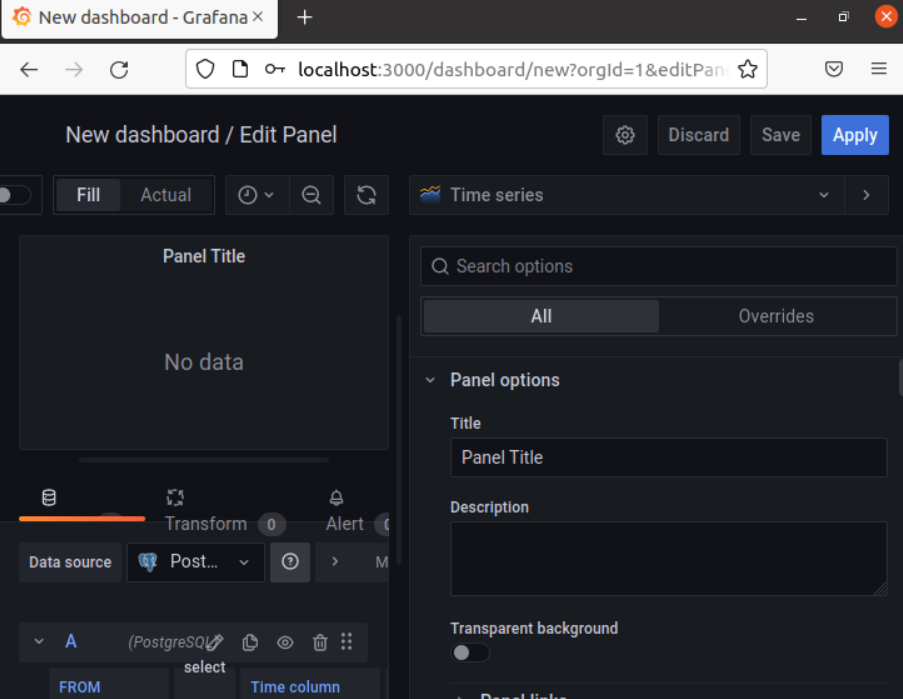
Теперь можно заняться созданием дашборда, который будет отображать информацию, полученную из базы данных. Для этого в Grafana надо воспользоваться вкладкой “Create Dashboards” (создать дашборд)



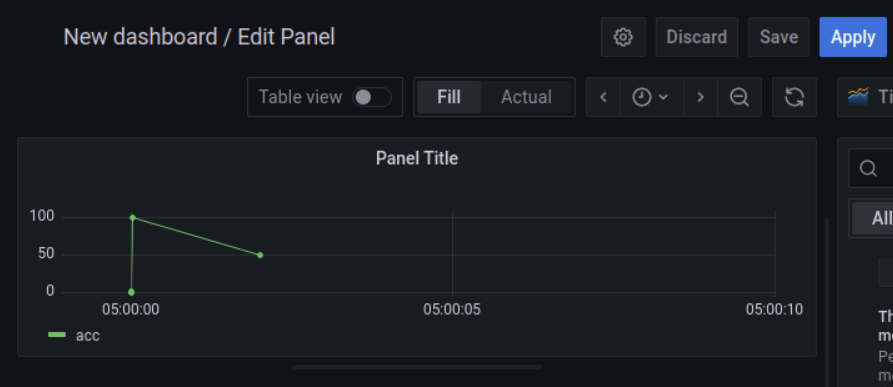
в предложенном интерфейсе будут предложены несколько опций для создания графиков



давайте создадим новую панель, нажав на клавишу “Add a new panel”, после чего вы увидите форму для редактирования данных дашборда.



Наиболее важным разделом является “Data source” (источник данных), в котором вы задаете способ получения данных для отображения. Вам необходимо указать столбец базы данных, который играет роль временной метки, это будет абсцисса графиков. Это обязательно, так как grafana отображает информацию в привязке ко времени. Кроме этого необходимо указать столбец который будет отображаться. После этого необходимо нажать на кнопку “Apply” (применить) и вы увидите на дашборде значения величины, сохраненной в базу данных



У этого дашборда можно менять названия, подписи, цвет и тип линии и многое другое. Это делается через простой и понятный интерфейс, предлагаем вам самостоятельно поэкспериментировать с параметрами. Также вы можете установить период обновления запросов в базу данных, из которой берутся данные для отображения. Установив этот таймер, например, на 5 секунд, и внеся изменения в базу данных, вы увидите эти изменения на дашборде через 5 секунд.

*Тест:*

1. Для решения каких задач может использоваться Grafana? (0.25)
   1. обучение моделей машинного обучения
   2. **визуализация процессов**
   3. **мониторинг и контроль процессов**
   4. **отправка оповещения о срабатывании триггера**
2. Какой порт по умолчанию используется в web-интерфейсе grafana? (0.25)
   1. 1000
   2. 2000
   3. **3000**
   4. 4000
3. С какими источниками данных может быть состыкована Grafana? (0.25)
   1. **реляционная база postgresql**
   2. **база данных временных рядов terminusDB**
   3. **реляционная база данных mysql**
   4. **prometeus**
4. Какие сущности используются в grafana для визуализации? (0.25)
   1. **панель**
   2. линейка
   3. **дашборд**
   4. **аннотация**

*Итоги/выводы*

В этом юните вы узнали про инструмент Grafana и для организации мониторинга и контроля. Вы научились его устанавливать, проводить настройку и создавать простые дашборды для визуализации информации. В следующем юните мы применим полученные знания для проекта машинного обучения.

# Модуль 8. Юнит 4. Практический пример использования Grafana для проекта машинного обучения.

*Введение:*

В предыдущем юните вы познакомились с установкой и настройкой Grafana, а также с созданием простейших интерактивных дашбордов. В этом юните вы примените полученные знания для использования в проекте машинного обучения.

*Содержание юнита:*

Как видите, настройка Grafana для работы очень проста. Применение для мониторинга работы системы заключается в том, что исполняемый скрипт или программа пишет в специальную базу данных информацию о важных параметрах процесса, а специалист эксплуатации наблюдает за этими параметрами на дашборде, либо может установить правило срабатывания аварийного сигнала (Alert) при превышении некоторого порогового значения. Этот же подход используется в проектах машинного обучения. В различных процессах, которые входят в конвейер машинного обучения, исполняемый скрипт или главная программа записывают в базу данных важную для мониторинга и контроля информацию, а Grafana читает эту информацию и отображает на специально подготовленных дашбордах.

В комплексных проектах, состоящих из множества отдельных компонентов, Grafana может запускаться в виде микросервиса в общей инфраструктуре программной системы. Вот один из примеров рабочей архитектуры проекта машинного обучения, включающего модуль Grafana для мониторинга и визуализации

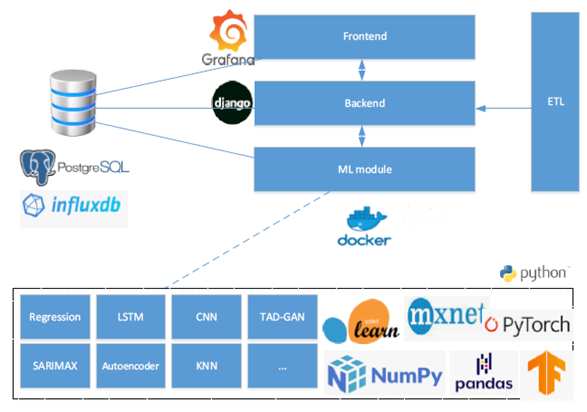


Рисунок “Архитектура проекта машинного обучения с использованием Grafana”

В этом проекте в виде отдельных микросервисов с помощью контейнеров docker реализованы:

* подсистема сбора и обработки данных (ETL)
* backend подсистема на django
* базы данных (например, PostgreSQL и InfluxDB)
* микросервис модели машинного обучения с необходимыми библиотеками
* подсистема frontend, включая визуализацию с Grafana

backend на django организует работу всех остальных микросервисов, обеспечивает выполнение модели машинного обучения, сохранение в базу данных для последующего отображения этих данных в Grafana. Для управления такими многоконтейнерными приложениями обычно используют docker-compose или kubernetes.

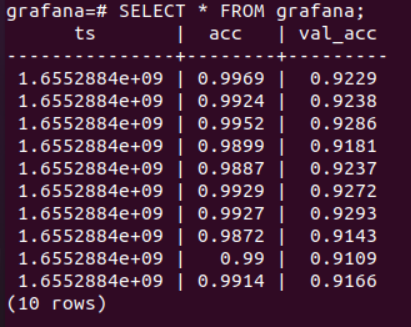
С созданием многоконтейнерных приложений вы познакомитесь в следующих модулях курса, а сейчас давайте реализуем запись параметров обучения модели на простом примере в базу данных PostgreSQL и ее отображение в Grafana.

Для подключения к PostgreSQL в python используется библиотека psycopg2. С ее помощью вы можете присоединиться к базе данных и записывать в нее необходимую информацию.

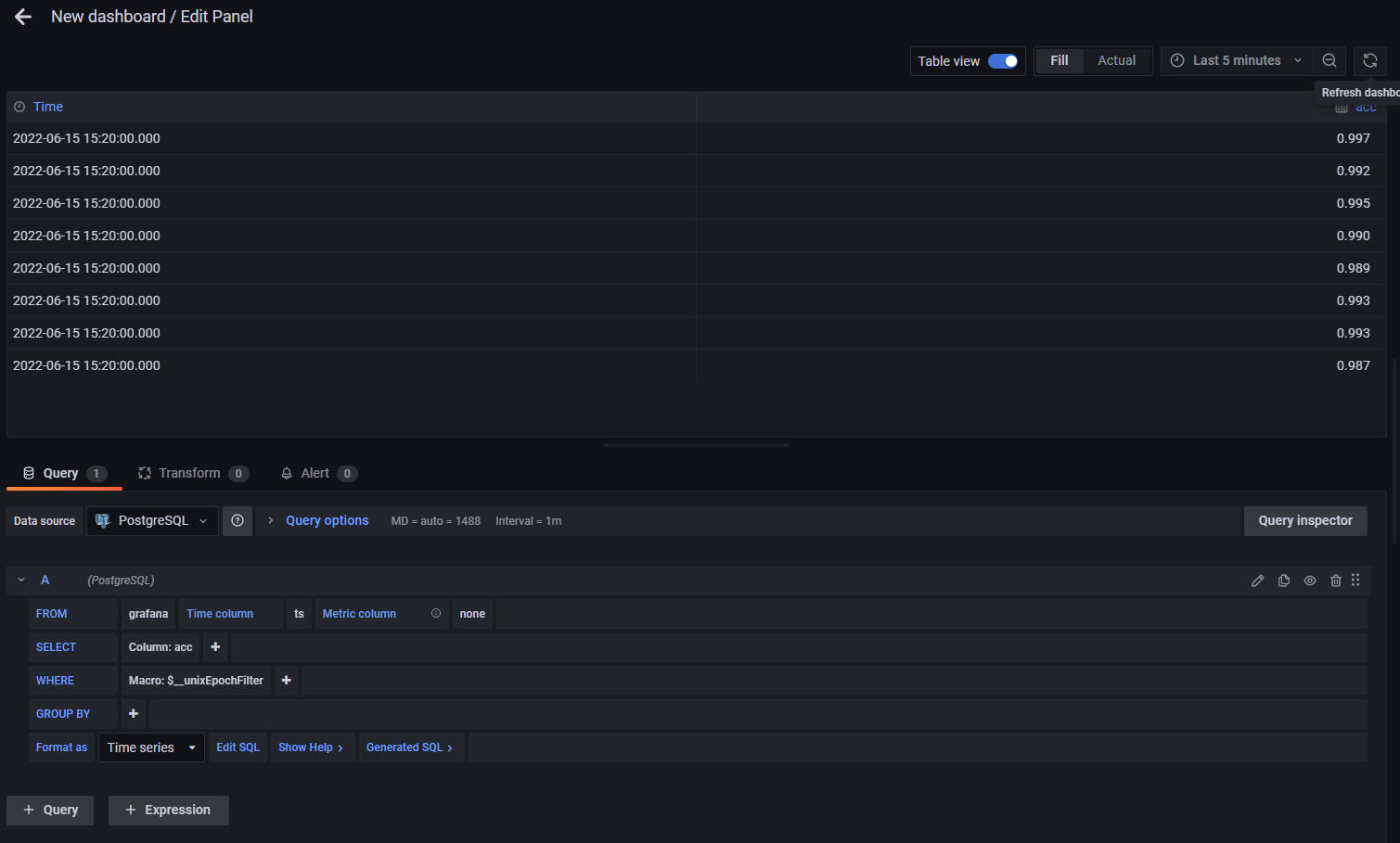
Давайте используем базу данных, созданную в предыдущем юните, чтобы для каждого этапа обучения модели записать timestamp, accuracy и validation\_accuracy



Выполнение этого кода приводит к тому, что в базе данных появляются строки в таблице



Соответственно эта информация считывается в Grafana из базы данных в табличном виде или в виде графика



*Практические задание:*

Выполните описанные в юните шаги самостоятельно.

Шаг 1: создайте виртуальную машину для PostgreSQL, создайте базу данных grafana, с использованием сведений из предыдущего юнита.

Шаг 2: создайте модель машинного обучения в jupyter ноутбуке

Шаг 3: создайте привязку к базе данных способом, изложенным в этом юните

Шаг 4: Запустите сервер Grafana с использованием информации из предыдущего юнита, сделайте привязку к таблице grafana базы данных PostgreSQL, созданной на шаге 1.

Шаг 5: Проведите обучение модели с записью accuracy и validation\_accuracy в базу данных.

Шаг 6: Проверьте, что в дашборде Grafana сохраненная информация нормально визуализируется.

*Итоги/выводы:*

В этом юните мы создали простейшую модель машинного обучения, провели ее обучение с записью параметров в базу данных и последующей их визуализацией с использованием Grafana.

# Итоги/выводы по модулю

В этом модуле, который завершает первый семестр обучения по данному курсу, были рассмотрены задачи и инструменты эксплуатации. Выход программной системы в эксплуатацию является важным закономерным этапом разработки и развития продукта. На этапе эксплуатации проверяется качество работы всей команды, правильность реализации алгоритмов и точность выполнения технического задания. От успешной эксплуатации зависит успех проекта целиком. На этом этапе выявляются ошибки, новые идеи.

Для проекта машинного обучения эксплуатация имеет особенное значение, поскольку это своего рода самое главное тестирование качества работы модели, на реальных данных, в реальных производственных условиях. Нередко самые перспективные гипотезы и удачные реализации моделей машинного обучения оказываются несостоятельными перед реальными данными или производственными условиями. Поэтому к процессу эксплуатации модели машинного обучения необходим такой же подход, что и в любом другом проекте разработки программного обеспечения.

Для эффективной эксплуатации придуманы и используются на практике различные организационные и технологические инструменты. В этом модуле мы сосредоточились на средствах мониторинга и контроля процессов в проекте машинного обучения. Сначала мы изучили консольные команды linux, поскольку это самая популярная операционная система для реализации проектов ML. Затем рассмотрели средство мониторинга Tensorboard, который позволяет осуществлять мониторинг и контроль параметров модели во время обучения. И, в последнем юните, мы рассмотрели более универсальный инструмент, фреймворк Grafana, который используется не только в проектах ML, но и вообще является популярным инструментом визуализации в проектах разработки программного обеспечения. Grafana обладает полезными функциями:

* имеет множество коннекторов для взаимодействия с различными источниками данных
* позволяет создавать наглядные дашборды, управлять параметрами визуализации
* имеет инструменты для контроля превышения пороговых значений
* позволяет отправлять уведомления при возникновении внештатных ситуаций.

Кроме рассмотренных в модуле инструментов существуют и другие средства для мониторинга и контроля. Обычно конкретное средство подбирается под требования проекта. Для небольших проектов достаточно консольных команд и небольших скриптов для автоматизации. В более сложных проектах не обойтись без таких инструментов как Grafana. Но и при использовании Grafana умение напрямую работать с системой через командную строку оказывается очень эффективным во многих ситуациях.

Полученные знания позволят вам самостоятельно спланировать и настроить систему эксплуатации для вашего проекта.

# Полезные ссылки

Ссылки на полезные книги по linux

https://tproger.ru/books/linux/

Описание инструмента Tensorboard на официальном сайте

<https://www.tensorflow.org/tensorboard>

Работа с SQL в PostgreSQL

https://www.digitalocean.com/community/tutorials/introduction-to-queries-postgresql