Spark overview

#checkup_1

1) Объектом исследования является «спрос» - задача прогноз спроса. Трудность в том что бы определить что называем спросом. Ограничивающий фактор (дефицит склада) – невозможно пробить это ограничение.
2) При оптимизации – мы должны по разному штрафовать за перепрогноз или недопрогноз. (недопрогноз это хуже чем перепрогноз)
3) Также необходимо определить множество признаков для модели. Сложность генерации признаков.
4) Необходимо решить вопрос с гранулярностью – как объединяем товары ? в группы категории Как разделяем регионы ? Какую задаем периодичность данных?

#checkup_2

1) как хранятся данный в parquet, csv, orc?

2) что такое Big Data? Как помогают решения Saas/PaaS/IaaS?

1) Как хранить неструктурированные данные?

- **NoSQL (non-relational) БД** хранит большие массивы данных без определенной структуры. Например, БД на основе пар «ключ-значение», документ (JSON).
- **SQL БД** жестко задает таблицы, строки, столбцы, индексы, отношения между таблицами;
- **data lakes** - озеро данных, хранилище, которое может содержать любые данные, в том числе структурированные, частично структурированные или неструктурированные из разных источников. Часто используется для хранения “сырых” данных, но может хранить и результаты анализа. Озера данных можно использовать для любых целей: анализов, прогнозов, оптимизации бизнес-процессов. Озера данных дешевле обычных баз данных, они более гибкие и легче масштабируются.
- **data warehouses (DWH)** - реляционная SQL база данных, специально разработанная и предназначенная для отчётов и бизнес-анализа. Существует несколько основных архитектур DWH (Kimball, Inmon, Data Vault). Данные, поступающие в хранилище данных, как правило, доступны только для чтения.
- Реляционные БД (предобработанные), Операционные БД (в реальном времени).

2) Data Scientist VS Data Engineer разница?

- **Data Engineer**: Разработка, построение и обслуживание инфраструктуры работы с данными;
- **Data Scientist**: Извлечение признаков из данных, Использование инструментов машинного обучения для прогнозирования и классификации паттернов в данных, Повышение производительности и точности алгоритмов машинного обучения и Проверка сформулированных гипотез.

3) В чем разница между Витриной данных (Datamart) и Data Warehouse?

- **Data warehouse** - система, которая агрегирует данные из нескольких источников в состоятельную БД для облегчения процессов AI/ML/Data mining.
- **Data mart** - специализированная (избранная) версия Data warehouse, которая содержит подмножество данных, которые нужны отдельно взятой команде. Данные берутся из Warehouse, Lake или иного источника.

4) Что такое shuffle? Для чего оно используется?

Метод, получающий на вход ряд (list, set) и меняющий порядок элементов в нём. Цель этого метода в получении случайной последовательности элементов - .

5) Как устроена хэш-функция в спарке?

Hash-function - любая функция, преобразующая входную строку случайного размера в значение фиксированного размера. В Spark типичная hash-function должна вычислить checksum (проверка на целостность данных) для упрощения процесса сравнения. НО может произойти коллизия при слишком большом кол-ве данных.

6) В чем разница между горизонтальным и вертикальным масштабированиями?

Вертикальное — это когда добавляют больше оперативки, дисков и т.д. на уже существующий сервер.

Горизонтальное — это когда ставят больше серверов в дата-центры, и сервера там уже как-то взаимодействуют.

7) Как посчитать что в следующий час выйдет из строя один сервер из 1000?

Пусть 2.5 в год каждый сервер простаивает (находится в поломке)

8) Когда стоит использовать распределенную систему, а когда не стоит?

Главная особенность распределенных систем – это высокая вероятность отказа и, хуже того, вероятность частичного отказа.
Если вы можете уместить задачу в памяти, то она тривиальна (не нуждается в множестве серверов и разделении задач).

9) Как обеспечивается репликация в HDFS? Для чего она нужна?

Зеркалирование и репликация осуществляются на уровне кластера, а не на уровне узлов данных.

Каждый файл делится на блоки дефолтного размера, при этом данные меньшего, чем 1 блок, размера сохраняются как “частичная запись”. Это осуществляет Namenode, которая и сохраняет отдельно мета-данные.

1 блок имеет отн-но большой размер (64мб, 128мб, 256мб), потому что данные слишком большие (террабайты) - блоков будет слишком много для быстрой обработки. А с блоками по 64-256 можно быстро осуществлять поиск.

А с учётом репликации (нужной для сохранения копии в случае shutdown, aka fault tolerance).

10) Должно ли быть число нод кратно какому-либо числу и если да то почему?

Мнение отдельных экспертов, что число нод в кластере Hadoop должно быть кратным заданному числу реплик в Hadoop. Вероятно, имеет смысл при малом числе нод 2-3-4. При большем их количестве скорее не важно. Ни в документации по системам Hadoop, ни в специальных обучающих курсах информация на эту тему отсутствует.

11) Сколько места занимает файл на 10мб в HDFS?

Файл размером 1 МБ хранится в блоке размером 128 МБ с 3 репликациями. Тогда файл будет храниться в 3 блоках и использовать только 3*1=3 МБ, а не 3*128=384 МБ. Но он показывает каждый размер блока как 128 МБ. Это просто абстракция для хранения метаданных в namenode, но не фактический размер используемой памяти.

Невозможно хранить больше файла в одном блоке. Каждый файл будет храниться в отдельном блоке.

12) Какие функции не сериализуются? Почему могут не сериализоваться?

- Non-serializable объект - сложный, вроде class или function. Это не серия, сугубо serializable объект (строка, число, пустое, бинарное значение).
- Serializable объект может быть трансформирован в другой вид типа текст, который можно легко далее передавать по процессам. Тогда как non-serializable нельзя.

#checkup_3

1) Виртуальное окружение чем отличается от виртуальной машины.

- **Виртуальное окружение** - это изолированное окружение среды (например Python в Anaconda), которое позволяет нам использовать определенные версии приложений.
- **Виртуальная машина** - это виртуальный компьютер, который использует выделенные ресурсы реального компьютера (процессор, диск, адаптер). Эти ресурсы хранятся в облаке и позволяют ВМ работать автономно.

2) Как изолируется python?

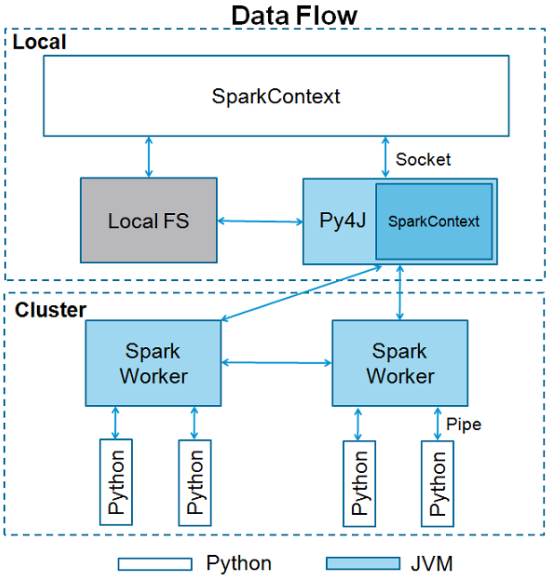
Виртуальное окружение Python создаётся путём копирования бинарного файла Python в локальную папку (ENV/bin/python). Родительская папка содержит символические ссылки на файлы стандартной библиотеки питона. Фактически, делаем свежую копию Питона на локальной системе.

3) Docker что это и зачем?

Docker - это программная платформа для быстрой разработки, тестирования и развертывания приложений. Docker *упаковывает ПО в стандартизованные блоки, которые называются контейнерами*. Каждый контейнер включает все необходимое для работы приложения: библиотеки, системные инструменты, код и среду исполнения. *Благодаря Docker можно быстро развертывать и масштабировать приложения в любой среде*.

3) SparkContext что это и зачем?

SparkContext – это точка входа в любую функциональность Spark. Когда мы запускаем любое приложение Spark, запускается программа драйвера, которая выполняет основную функцию, и здесь запускается SparkContext. Затем программа драйвера запускает операции внутри исполнителей на рабочих узлах (Spark Worker).



4) Когда лучше посчитать задачи локально, а когда отправить на кластер?

If I am in the same local network with the cluster, I would use the client mode and submit it from my laptop. If the cluster is far away, I would either submit locally with cluster mode.

Besides that Yarn-cluster mode makes sense for production jobs, while yarn-client mode (local) makes sense for interactive and debugging uses where you want to see your application's output immediately.

5) Что такое YARN?

Yarn is a package manager for your code. *It allows you to use and share (e.g. JavaScript) code with other developers from around the world.*

If you have problems, you can report issues or contribute back, and when the problem is fixed, you can use Yarn to keep it all up to date.

Code is shared through something called a package (sometimes referred to as a module). A package contains all the code being shared as well as a `package.json` file which describes the package.

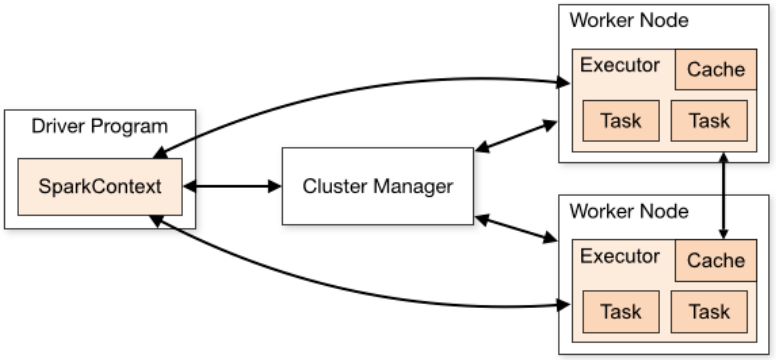
6) За что отвечает Spark driver? Что делает cluster manager?

Каждое Spark-приложение состоит из управляющего процесса – **драйвера (Driver)** – и набора распределённых рабочих процессов – **исполнителей (Executors)**.

Driver запускает метод `main()` нашего приложения. Здесь создаётся SparkContext. Обязанности **Spark Driver**:

- Запускает задание на узле в нашем кластере или на клиенте и планирует его выполнение с помощью менеджера кластера;
- Отвечает на пользовательскую программу или ввод;
- Анализирует, планирует и распределяет работу между исполнителями;
- Хранит метаданные о запущенном приложении и отображает в веб-интерфейсе.

Cluster manager (daemon) is usually backend GUI or CLI software that runs on a set of cluster nodes that it manages (in some cases it runs on a different server or cluster of management servers). The cluster manager works together with a cluster management agent. These agents run on each node of the cluster to manage and configure services, a set of services, or to manage and configure the complete cluster server itself.



7) Что такое демон-процесс (daemon)? Как он связан с мастером и воркерами?

Daemons are processes that run unattended. They are constantly in the background and are available at all times. Daemons are usually started when the system starts, and they run until the system stops. A daemon process typically performs system services and is available at all times to more than one task or user.

- The **master daemon** allocates resources across applications.
- The **worker daemon** monitors and reports resource availability and, when directed by the master, spawns executors. The worker also monitors the liveness and resource consumption of the executors.
  Other Spark daemon processes.

8) Где находится SparkContext?

SparkContext object (usually shown as `sc`) is the main entry point for Spark functionality and can be used to create Resilient Distributed Datasets (RDDs) on a cluster.

In Spark/PySpark you can get the current active SparkContext and its configuration settings by accessing `spark.sparkContext.getConf.getAll()`, here spark is an object of `SparkSession` and `getAll()` returns `Array[(String, String)]`.

9) Как происходит выполнение приложения Spark?

The components of the spark application are:

- **Driver** - converts a user application into smaller execution units called *tasks* and then schedules them to run with a cluster manager on executors. The driver is also responsible for executing the Spark application and returning the status/results to the user.
- **Application Master** - framework-specific entity charged with negotiating resources with ResourceManager(s) and working with NodeManager(s) to perform and monitor application tasks.
- **Spark Context** - allows Spark Driver to access the cluster through its Cluster Resource Manager and can be used to create RDDs, accumulators and broadcast variables on the cluster, also tracks executors in real-time by sending regular heartbeat messages.
- **Cluster Resource Manager (aka Cluster Manager)** - process that controls, governs, and reserves computing resources in the form of containers on the cluster.
- **Executors** - processes at the worker's nodes, whose job is to complete the assigned tasks. These tasks are executed on the worker nodes and then return the result to the Spark Driver.

Spark uses a master/slave architecture with a central coordinator called **Driver** and a set of executable workflows called **Executors** that are located at various nodes in the cluster.

10) Какие есть shared variables? В чем их отличие?

Normally, when a function passed to a Spark operation (such as `map` or `reduce`) is executed on a remote cluster node, it works on separate copies of all the variables used in the function. *These variables are copied to each machine, and no updates to the variables on the remote machine are propagated back to the driver program*. However, Spark does provide two limited types of *shared variables* for two common usage patterns: broadcast variables and accumulators.

- **Broadcast variables** allow the programmer to keep a *read-only* variable cached on each machine rather than shipping a copy of it with tasks. Broadcast variables are created from a variable `v` by calling `SparkContext.broadcast(v)`. The broadcast variable is a wrapper around `v`, and its value can be accessed by calling the `value` method.
- **Accumulators** are variables that are *only "added" to* through an associative and commutative operation and can therefore be efficiently supported in parallel. They can be used to implement counters (as in MapReduce) or sums. Spark natively supports accumulators of numeric types, and programmers can add support for new types. A numeric accumulator can be created by calling `SparkContext.longAccumulator()` or `SparkContext.doubleAccumulator()` to accumulate values of type Long or Double, respectively.

11) Что такое broadcast join?

Spark SQL uses **broadcast join** (aka **broadcast hash join**) instead of hash join to optimize join queries when the size of one side data is below spark.sql.autoBroadcastJoinThreshold (10M rows).

Broadcast join can be very efficient for joins between a large table (fact) with relatively small tables (dimensions) that could then be used to perform a **star-schema join**. It can avoid sending all data of the large table over the network.

```
from pyspark.sql.functions import broadcast
data1.join(broadcast(data2), data1.id == data2.id)
```

12) Что такое оконная функция и как она работает?

Оконная функция в SQL (Window function) - функция, которая работает с выделенным набором строк (окном, партицией) и выполняет вычисление для этого набора строк в отдельном столбце. Это не то же самое, что GROUP BY. Они не уменьшают количество строк, а возвращают столько же значений, сколько получили на вход. Во-вторых, в отличие от GROUP BY, OVER может обращаться к другим строкам. И в-третьих, они могут считать скользящие средние и кумулятивные суммы.
Example

Партиции (окна из набора строк) - это набор строк, указанный для оконной функции по одному из столбцов или группе столбцов таблицы. Партиции для каждой оконной функции в запросе могут быть разделены по различным колонкам таблицы.
Habr

13) Нужна ли сортировка в оконных функциях?

Множество оконных функций можно разделять на 3 класса:

- Агрегирующие (Aggregate)
- Ранжирующие (Ranking) - необходим ORDER BY
- Функции смещения (Value)

14) Что такое лик в данных, что недообучение и переобучение?

Data Leakage - when the data you are using to train a machine learning algorithm happens to have the information you are trying to predict.

- **Bias:** Assumptions made by a model to make a function easier to learn. It is actually the error rate of the training data. When the error rate has a high value, we call it High Bias and when the error rate has a low value, we call it low Bias.
- **Variance:** The error rate of the testing data is called variance. When the error rate has a high value, we call it High variance and when the error rate has a low value, we call it Low variance.

Underfitting: A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data, i.e., it only performs well on training data but performs poorly on testing data. (It's just like trying to fit undersized pants!). *High variance and low bias.*

Overfitting: A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. *Low variance and high bias.*

Pictures

15) Lag и Lead в чем различие?

- **lag()** - функция, возвращающая предыдущее значение столбца по порядку сортировки;

- **lead()** - функция, возвращающая следующее значение столбца по порядку сортировки.

16) Как посчитать среднее в разрезе категория + штат?

```
select cat_id, state_id,
--average grouped by category and state; no lines are excluded
avg(sales) over (partition by cat_id, state_id) as avg_sales,
from df_prices
```

17) Зачем нужен rangeBetween и как установить ограничение на один ряд вперед?

`pyspark.sql.Window.rangeBetween(start, end)`

Both `start` and `end` are inclusive. This function allows to calculate the 1st order difference: daily/weekly/monthly lag.

#checkup_4

1) В чем разница groupby и window functions? Когда использовать одно и когда другое?

#checkup_3, q8

2) Как работает условие в spark? When+otherwise?

```
import pyspark.sql.functions as F
#1 F.when(
#2 F.isnull(F.col('COLUMN_NAME')), 'value if True')
#3 .otherwise('value if False')
```

Устанавливается некоторая проверка с результатом Bool, если True → 'value if True', иначе (буквально .otherwise) 'value if False'.

3) Что такое винсоризация и как это реализовать в spark?

Winsorizing or **winsorization** is the transformation of statistics by limiting extreme values in the statistical data to reduce the effect of possibly spurious outliers.

The distribution of many statistics can be heavily influenced by outliers. A typical strategy is to set all outliers to a specified percentile of the data; for example, a 90% winsorization would see all data below the 5th percentile set to the 5th percentile, and data above the 95th percentile set to the 95th percentile. Winsorized estimators are usually more robust to outliers than their more standard forms.

```
#create boundary values at 10th  and 90th percentile: 80% winsorization
df = (
    df
    .withColumn('10_percentile', F.percentile_approx("sales",
lower_percentile, 1000000).over(window_all_df))
    .withColumn('90_percentile', F.percentile_approx("sales",
higher_percentile, 1000000).over(window_all_df))
  )
#apply the newly created boundaries to the existing values in 'sales'
  df  = (
      df
      .withColumn('sales_visorized',
                  F.when((F.col('sales') < F.col('90_percentile'))
                  &
                  (F.col('sales') > F.col('10_percentile')),
                  F.col('sales'))
                  )
  )
```

4) Как взять окно по диапазону в днях? (Подсказка: rangebetween + seconds)

a) Convert 1D to seconds:

```
days = lambda i: i * 86400
```

b) Take the date from the DB, convert to `"timestamp"` and apply days function in `rangeBetween`:

```
window_agg = (Window
              ##some code##
```

```
.orderBy(
    F.col("date").cast("timestamp").cast("long")
    )
.rangeBetween(-days(8), -days(1)) )
```

5) Разница null vs nan (once again)

- Nan - non a number
- Null - empty set (∅)

6) Зачем нужен бейзлайн алгоритм?

A baseline model is essentially a simple model that acts as a reference in a machine learning project. Its main function is to contextualize the results of trained models.

Baseline models usually lack complexity and may have little predictive power. Regardless, their inclusion is a necessity for many reasons.

1. Baseline models serve as benchmarks for trained models
   You can support a model if it addresses the flaws of the baseline and displays a stronger overall performance. If the trained model does not perform better than the baseline model, it means that the model's added complexity does not provide enough benefit. Incorporating baseline models mitigates the risk of adding unneeded model complexity.
2. Baseline models improve understanding of your data
   Furthermore, if your trained models are unable to outperform the baseline model, it could be a sign that the data set lacks predictive power. In such a case, it would be a good idea to verify the quality of the data instead of continuing to train models.

7) Как считается MAPE и зачем она нужна?

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

В чём прелесть такой формулы: очень легко интерпретировать. Мы ищем отклонения в долях (читай %) от истинного значения.

В отличие от MAE и MSE мы оперируем относительной величиной, а не абсолютной.

8) Как строить трейн/тест/валидацию для временных рядов?

В TS есть зависимость между наблюдениями вплоть до сезонности, поэтому случайное разбиение **не подойдёт**. Недавние данные для test, Более ранние для train.

For example, if you had 144 records at monthly intervals (12 years), a good approach would be to keep the first 120 records (10 years) for training and the last 24 records (2 years) for testing. tds

9) Что такое VectorAssembler в spark и зачем он нужен?

VectorAssembler получает на вход n векторов и объединяет их в один большой вектор:

```
VectorAssembler(inputCols=FEATURES_COL, outputCol='new_merged_col')
```

Какой-то фундаментальной причины для такого объединения нет. Просто это такой способ задавать вектор объясняющих переменных для i-го наблюдения.

10) Что такое Pipeline в spark?

```
#create x's and y
assembler = [VectorAssembler(inputCols=FEATURES_COL,
outputCol='features')]
#fit linear regression
```

```
lr = [LinearRegression(featuresCol='features', labelCol=LABEL_COL,
maxIter=10)]
stages = assembler + lr #must be list
p = Pipeline(stages=stages) #pipeline contains list of stages to
execute
```

11) Как разделить DataFrame на две выборки в отношении X и Y?

`df.randomSplit(weights = [X, Y], seed = 42),`

`weights` – доли обучающей и тестовой выборок (если сумма не равна 1, нормируются),

`seed` (опционально) - случайное целое число, которое инициализирует random generator.

12) Как от больших данных перейти к малым и провести базой анализ?

Можно сделать небольшую выборку, например из 10-50к данных – обычно этого достаточно чтобы построить распределения и провести простой EDA.

13) Как делать перебор параметров и GridSearch в Spark?

We use a ParamGridBuilder to construct a grid of parameters to search over. With 3 values for 1st step of *Pipeline* and 2 values for lr.regParam (linear or logistic), this grid will have 3 x 2 = 6 parameter settings for CrossValidator to choose from.

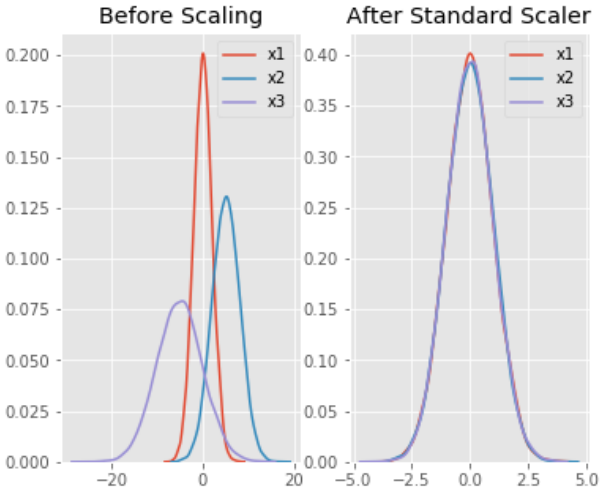
```
paramGrid = ParamGridBuilder() \
    .addGrid(pipeline_n.numFeatures, [10, 100, 1000]) \ #first
    .addGrid(lr.regParam, [0.1, 0.01]) \ #second
    .build()
```

Link

14) StandardScaler VS MinMaxScaler в чем разница и когда какой применять?

- `StandardScaler` - масштабирует аналогично нормировке случайной величины:
  `z = (x - mu)/std`
- `MinMaxScaler` - нормирует каждое наблюдение на величину разброса наблюдений:
  `z = x/[max(X) - min(X)]`

StandardScaler is useful for the features that follow a Normal distribution. This is clearly illustrated in the image below (source).



MinMaxScaler may be used when the upper and lower boundaries are well known from domain knowledge (e.g. pixel intensities that go from 0 to 255 in the RGB colour range).

15) Как задать pipeline в spark для Логистической регрессии?

```
from pyspark.ml.classification import LogisticRegression
```

```
assembler = [VectorAssembler(inputCols=FEATURES_COL,
outputCol='features')]
lr = [LogisticRegression(featuresCol='features', labelCol=LABEL_COL,
maxIter=10)]
stages = assembler + lr
p = Pipeline(stages=stages)
# + train, test, scaler
```

Вопросы с собеседований:

1) Оптимальные константы для MSE и MAE? Запрограммируйте эти метрики в spark.

```
from pyspark.ml.evaluation import RegressionEvaluator
# set the parameters of the evaluator
evaluator = RegressionEvaluator(predictionCol="prediction",
labelCol="actual_y", metricName="metric_name")
# apply the config to your data
evaluator.evaluate(datatable.withColumn("prediction", "actual_y")
```

Spark doc

Сами метрики по абсолютному значению в зависимости от данных сильно разнятся. Поэтому если вспомнить про baseline, то бОльшая разница с baseline - критерий в пользу улучшения модели. Оптимальное значение находится в области, когда train и test дают соизмеримые результаты.

2) Как по-разному штрафовать за overfitting и underfitting?

- Overfitting - L1, L2 регуляризация (штраф за избыточные веса)
- Underfitting - для недообучения рекомендуют 1)дольше учить модель, 2)собрать больше данных, 3)больше эпох + shuffle после каждой эпохи

3) Что такое intercept в Линейной регрессии?

Значение объясняемой переменной при равенстве нулю всех объясняющих переменных. Аналитически - это т. пересечения с осью ординат.

4) Что такое точность? Что такое полнота?

Коротко: `Precision` можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а `recall` показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

Более подробно с примером: Habr

5) Какие критерии используются для разбиения в узлах дерева?

- `criterion{“gini”, “entropy”, “log_loss”}` - loss-функция, которую нужно минимизировать;
- `max_depth` - максимальная глубина дерева;
- `min_samples_split or min_samples_leaf` - ограничения по кол-ву элементов для разбиения или нахождению в одной из нод.

#checkup_6

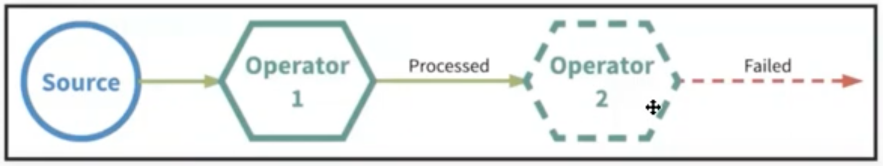
1) Какие типы падений (failure) есть в streaming? Чем отличается отказ во время обработки от отказа отправки ответа о результатах?

Типы failures:

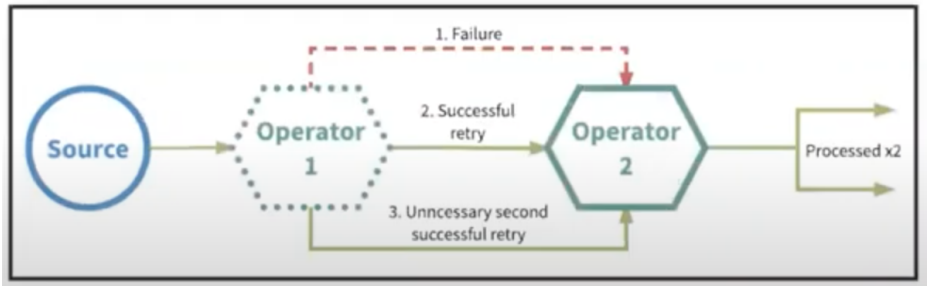
1) Не дошло до следующего Этапа и ничего не произошло;

2) Повторно отправляем запрос - всё ок, но не получаем ответа (т.е как будто ничего не произошло), и ещё раз отправляем - задваиваем обработку.

- отказ во время обработки: “поручение” из Этапа 1 **просто не дошло** до Этапа 2. Более простая схема без обратного хода:



- отказ отправки ответа о результатах: Этап 1 не знает о результате отправки “поручения” Этапу 2 и в независимости от исхода (успех/провал) отправляет повторно “поручения” - риск задваивания результата как на фото ниже:



2) Подходы для обработки потоковых данных? Обработка батчами против обработки на лету?

1) **Batch Processing** - обработка происходит по прошествии некоторого времени по накопленным за это время данным (Например, *ежемесячные* счета за свет, воды)
2) **Real Time Processing** - обработка происходит практически мгновенно по мере поступления новых данных (Например, банковские переводы через СБП (Россия), QuickPay (Кипр))

3) Что такое лямбда архитектура и почему ее удобно реализовывать в Spark?

На практике, некоторые операции, в частности, классический MapReduce на Hadoop, могут выполняться достаточно долго. В ряде случаев подобная задержка приведет к устареванию информации и потери ее актуальности.

Для этого была предложена **лямбда-архитектура**, когда обработка данных разделяется на 2 пути:

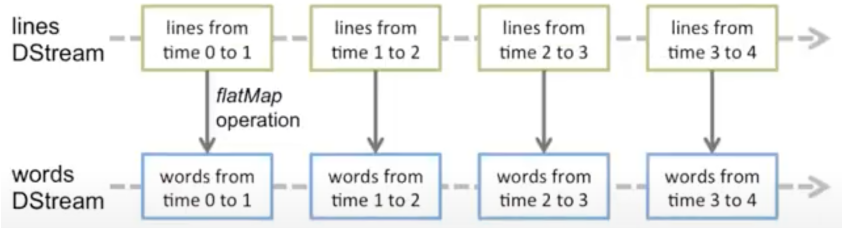
- «холодный путь» — пакетный уровень, где все входящие данные хранятся в необработанном виде и обрабатываются в пакетном (batch) режиме;
- «горячий путь» — скоростной уровень (или уровень ускорения), где данные анализируются в режиме реального времени. Этот уровень обеспечивает минимальную задержку обработки с некоторой потерей точности.

Итого: lambda – это архитектурный подход, при котором произвольная функция применяется к произвольному набору данных за минимальное время.

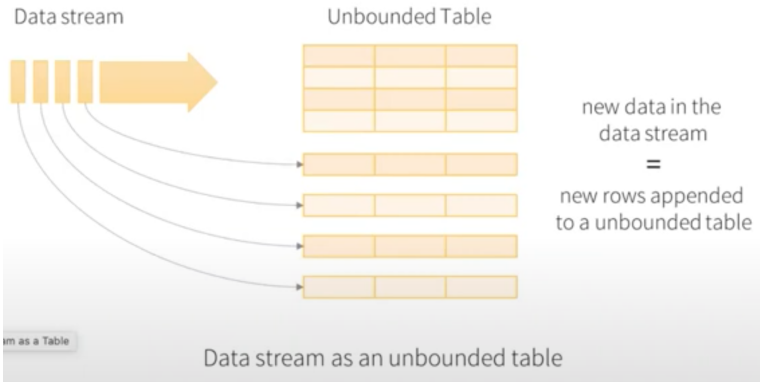
Применение в Spark удобно за счёт унификации кода для обоих процессов lambda (один код для real time & batch processing).

4) Как спарк представляет поток данных в таблицы? В чем разница Dstream от Structured dataframe?

- **Dstream (Discretized Stream)** - непрерывный поток поступающих данных, которые обрабатываются по batch, а не скопом;



- **Structured dataframe** - единая таблица (“монолит”), к которой добавляются новые строки.



5) Как работает Dstream в Spark? Как выполняется операция Map для стриминга?

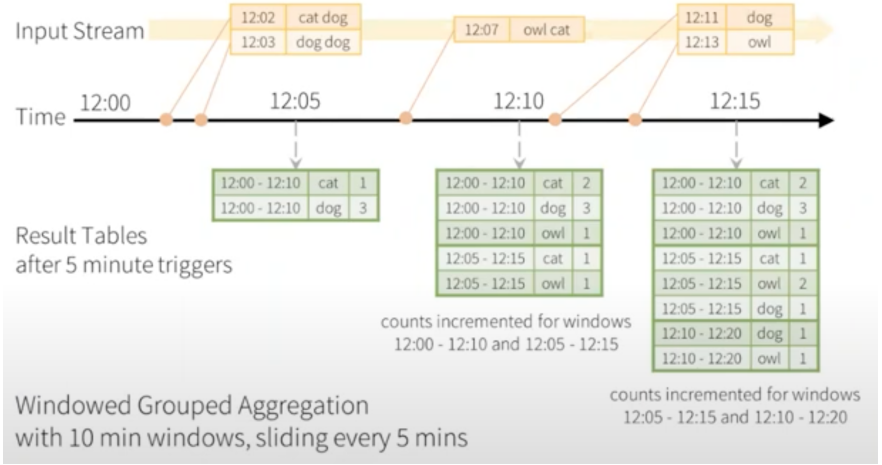
Dstream: входные данные разбиваются на batches, обрабатываются (Map-Reduce etc.) - получаются блоки в виде RDD (an immutable distributed collection of elements of your data, partitioned across nodes in your cluster that *can be operated in parallel*).

Map: предварительная обработка входных данных в виде большого список значений. При этом главный узел кластера (master node) получает этот список, делит его на части и передает рабочим узлам (worker node). Далее каждый рабочий узел применяет функцию Map к локальным данным и записывает результат в формате «ключ-значение» во временное хранилище.

6) В Structured Streaming dataframe чем отличается временное окно для триггера от временно окна обработки?

Триггер - каждые X мин/с/ч *обновить* выходящую таблицу, в которой результаты разбиваются по Временному окну.

Временное окно - агрегирующий диапазон (диапазон длиной Y минут, например) в выходящей статистике, которая считается, скорее всего, реже, чем вызывается триггер.



Интересное про Spark: если случился failure и данные пришли позже ожидаемого, Spark сможет идентифицировать “корректный” timeframe для конечной статистики.

7) Что такое Kafka? Что такое topic в kafka?

Kafka - распределённый программный брокер сообщений*

***Брокер сообщений** представляет собой тип построения архитектуры, при котором элементы системы «общаются» друг с другом с помощью посредника. Благодаря его работе происходит снятие нагрузки с веб-сервисов, так как им не приходится заниматься пересылкой сообщений: всю сопутствующую этому процессу работу он берёт на себя.

Topics: Events are organized and durably stored in topics. Very simplified, a topic is similar to a folder in a filesystem, and the events are the files in that folder. An example topic name could be "payments".

8) По каким портам подключаются обычно jupyter, spark ui?

- Spark UI: 4040 port
- Jupyter: 8888 port
- SSH: 23 port

9) Что такое Streaming Spark context? За что отвечает флаг StopGracefully?

Streaming Spark context - main entry point for Spark Streaming functionality. It provides methods used to create DStreams from various input sources. It can be either created by providing a Spark master URL and an appName or from existing SparkContext() etc.

stopGracefully: bool, optional

Stop gracefully by waiting for the processing of all received data to be completed

10) Что такое zookeeper?

ZooKeeper хранит самые важные метаданные системы: где находятся партиции, кто из реплик лидер и т.д. По сути, это что-то вроде специфической файловой системы или API триггеров поверх согласованного лога. А Kafka — это API pub/sub (издатель-подписчик) поверх согласованного лога.

Обновление от 2021: Если запустить Kafka с новым контроллеров кворума, все задачи по метаданным, которые раньше выполнялись контроллером Kafka и ZooKeeper, ложатся на новый сервис, запущенный прямо в кластере Kafka. Контроллер кворума можно запустить на выделенном оборудовании, если этого требует ваш сценарий.

#checkup_7

1) Как настроить поток из файла с помощью Spark Streaming?

```
(
    input_stream
#split the input
    .withColumn("token",functions.split("site", "\."))
.withColumn("token",functions.col("token").getItem(functions.size("toke
n") - 1))
#group/count etc.
    .groupBy("token")
    .count()
    .orderBy(functions.desc("count"))
#add streaming part
    .writeStream
    .trigger(processingTime="30 seconds") #parse data every 30s
    .outputMode("complete")
    .format("memory")
    .queryName("zones") #name of the output
    .start()
)
```

2) Как задать pipeline для построения модели текстовой классификации в спарк на NaiveBayes?

This example shows extensive description of the **NaiveBayes** realisation in spark and an official spark example*.

* the files can be loaded also as below:

```
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load("file_path")
```

Pipeline example from Spark class:

```
main_pipe = pipeline.Pipeline(
    stages=(
        feature.Tokenizer(inputCol="message", outputCol="tokens"),
        feature.CountVectorizer(
            maxDF=4000,
            minDF=5,
            inputCol="tokens",
            outputCol="vector",
        ),
        feature.StringIndexer(inputCol="label", outputCol="y"),
        classification.NaiveBayes(featuresCol="vector", labelCol="y"),
    )
)
```

3) Как задается schema в спарк для формирования таблицы? (types.StructType())

Spark Schema defines the structure of the DataFrame which you can get by calling `printSchema()` method on the DataFrame object. We can define it using `StructType` class which is a collection of `StructField` that define the column name (String), column type (DataType), nullable column (Boolean) and metadata (MetaData).

Spark provides `spark.sql.types.StructType` class to define the structure of the DataFrame and It is a collection or list on StructField objects.

```
struct1 = StructType([StructField("f1", StringType(), True)])
>>> struct1["f1"]
StructField('f1', StringType(), True)
>>> struct1[0]
StructField('f1', StringType(), True)
```

4) Какая $bash команда записывает строки в файл? [1]

```
# Define the filename
filename='books.txt'
# Type the text that you want to append
read -p "Enter the text that you want to append:" newtext
# Append the text by using '>>' symbol
echo $newtext >> $filename
```

From the Spark class:

```
#define the directory for the input
input_stream_dir = "/stream/structured/"
#name the file
!mkdir -p $input_stream_dir
#read and write X lines and add to the tempfile
!shuf top-1m.csv | head -1000 > `tempfile -d $input_stream_dir`
```

5) Как выглядит использование Spark Streaming для построения дашборда? (Подсказка: есть привязка к файлу, раз в t секунд запускается логика спарка + формируется некоторая SQL таблица)

1) Есть обновляемый файл. В учебном формате можно “вручную” добавлять к нему по X записей:
   `!shuf file_name.csv | head -1000 > `tempfile -d $input_stream_dir``
2) Для него запускается streaming по структуре из вопроса 1 (#checkup_7);
3) Итог записывается в некое подобие SQL таблицы, которую уже можно “читать”:
   `spark.sql("select * from zones").show()`

6) Можно ли джойнить статичную таблицу и стриминговую таблицу?

Да, загружаем оба файла, задаем schema. Для обновляемого запускаем streaming и отдельно прописываем join:

```
table1 = table1.join(
    table2,
    on=(table1.site == table2.site)
)
```

7) В чем отличие, если мы делаем джойн двух стриминговых таблиц или стриминговой и статичной?

В данном случае запуск streaming и join находятся в одном процессе для второй таблицы:

```
### first file’s streaming ###
    --- // ---
```

```
    ### second file's streaming ###
    (
  table2
  .join(table2, on="site")
  .writeStream
  .format("memory")
  .trigger(processingTime="5 seconds")
  .queryName("stream_stream")
  .start()
)
```

8) Как установить периодичность обновления витрины данных в Spark Streaming? [1, 2]

- For spark Real-time Streaming, you have to create a StreamingContext, which serves as the primary entry point. Import the necessary packages:
  `from pyspark import SparkContext`
  `from pyspark.streaming import StreamingContext`
- Define the batch duration while constructing a StreamingContext. For example, here, we provided the batch duration as 6 seconds:
  `sc = SparkContext(appName = "new app")`
  `strc = StreamingContext(sc, 6)`
- Now one can begin receiving data in the form of DStreams via the TCP protocol on a specified port. For example, in our case, the hostname is "localhost," and the port is 8084 (Jupyter: 8888):
  `text_data = strc.socketTextStream("localhost", 8084)`
- Then one can write some function to process the data;
- To begin the Spark Real-time Streaming process and continue receiving real-time streaming data, use the start() function with the StreamingContext object, i.e., "strc."
  `strc.start()`
- The data will be streamed until the **awaitTermination()** method gets a termination instruction, such as (Ctrl+C or Ctrl+Z). Now you have to send the previously streamed text data *from the data server to the spark streaming server*.
- In order to transfer the text data from the data server to the spark streaming server, we must run the 'nc' command in the Netcat Utility. Netcat is an application that allows you to read and write to network connections via TCP or UDP ports.
  `nc -lk 8083`
- In the above command, -l permits "nc" to listen for incoming connections rather than connecting to a remote host, while -k enables "nc" to keep listening for new connections after the current one is finished.
- Now, execute the following command to implement text cleaning on the received data on port 8083.
  `spark-submit streaming.py localhost 8083`
- After executing the above command for Spark Real-time Streaming, any data received in the netcat server is processed, and the processed content is printed in another terminal every 6 seconds, which is the batch duration provided while creating the streaming context.

#checkup_8

1) Как настроить SparkStreaming для топика kafka?

```
sc = pyspark.SparkContext()
ssc = StreamingContext(sc,5)

kafka_topic_name = "reco-train"
kafka_bootstrap_servers = 'localhost:9092'

kvs = KafkaUtils.createStream(ssc, kafka_bootstrap_servers,
'spark-streaming-consumer', {kafka_topic_name:1})
kvs = KafkaUtils.createDirectStream(ssc, [kafka_topic_name],
{"metadata.broker.list": kafka_bootstrap_servers})
kvs = KafkaUtils.createDirectStream(ssc, [kafka_topic_name], {
                        'bootstrap.servers':kafka_bootstrap_servers,
                        'group.id':'test-group',
                        'auto.offset.reset':'largest'})
### some data processing (in the case below it is word counter) ###
lines = kvs.map(lambda x: x[1])
counts = lines.flatMap(lambda line: line.split(' '))
counts = lines.flatMap(lambda line: line.split(' ')).map(lambda word:
(word, 1)).reduceByKey(lambda a, b: a+b)
counts.pprint()
### processing step end ###
ssc.start()
ssc.awaitTerminationOrTimeout(50) # stream will run for 50 sec
ssc.stop()
sc.stop()
```

2) Зачем нужен pprint() перед запуском spark streaming context?

Для вывода результата в streaming нужно применить ф-ию вывода (std. out). В данном случае - это .pprint()

3) Как настроить countByValueAndWindow за что отвечают настраиваемые параметры?

`DStream.countByValueAndWindow(windowDuration, slideDuration, numPartitions=None)`

- `windowDuration` - width of the window; must be a multiple of this DStream's batching interval;
- `slideDuration` - sliding interval of the window (i.e., the interval after which the new DStream will generate RDDs);
- `numPartitions` - number of partitions of each RDD in the new DStream.

4) Как настроить преобразование map над потоком из файла?

Пример с лекции:

```
from pyspark import streaming
sc = pyspark.SparkContext()
ssc = streaming.StreamingContext(sc, batchDuration=10) #start streaming
```

```
stream_input_dir = "./stream/sparkstreaming/" # streaming folder
!mkdir -p $stream_input_dir # put streaming input file in the above
input_stream = (
    ssc.textFileStream(directory=stream_input_dir)
) # activate streaming from input
# apply .map
odd_even_stream = (
    input_stream
    .map(int) #assign var type to the file
    .map(
        lambda x: "нечетное" if x % 2 else "четное"
    )
)
# run some aggregation function on the mapped result
```

5) Что такое udf в pyspark и когда ее использовать?

Option 1

```
import pyspark.sql.functions as f
# create py function
def udf_upper_names(x: str) -> str:
  step_2 = x.split()
  return ' '.join([word[0].upper() + word[1:] for word in step_2])
# apply the function converted to the UDF using f.udf from spark
udf_upper_namesUDF = f.udf(lambda z: udf_upper_names(z))
```

Option 2

```
@udf(returnType=StringType()) #annotation (preprocessing) ->
def upperCase(str):
      return str.upper()
```

6) Что делает explode и чем отличается от posexplode?

Spark function `explode(e: Column)` is used to explode or create array or map columns to rows. When an array is passed to this function, it creates a new default column “col1” and it contains all array elements.

`posexplode(e: Column)` creates a row for each element in the array and creates two columns “pos’ to hold the position of the array element and the ‘col’ to hold the actual array value. And when the input column is a map, posexplode function creates 3 columns “pos” to hold the position of the map element, “key” and “value” columns.

7) Что такое декоратор? Зачем он нужен?

Декораторы — это, по сути, "обёртки", которые дают нам возможность изменить поведение функции, не изменяя её код.

8) Какой тип считывания в спарке позволяет работать с картинками?

```
spark.read.format("image").load(images_dir).filter("image.nChannels >
int AND image.height < int")
```

9) Какие есть атрибуты у картинок по дефолту доступные в спарке?

- origin: `StringType` (represents the file path of the image)
- height: `IntegerType` (height of the image)

- width: `IntegerType` (width of the image)
- nChannels: `IntegerType` (number of image channels)
- mode: `IntegerType` (OpenCV*-compatible type)
- data: `BinaryType` (Image bytes in OpenCV-compatible order: row-wise BGR in most cases)

***OpenCV** is the huge open-source library for computer vision, machine learning, and image processing. OpenCV offers support for the image formats Windows bitmap (bmp), portable image formats (pbm, pgm, ppm) and Sun raster (sr, ras).

10) Image.Resampling.BICUBIC — что делает?

Bicubic interpolation is a 2D system of using cubic splines or other polynomial technique for sharpening and enlarging digital images. When we interpolate an image, we are actually distorting the pixels from one grid to another.

`PIL.Image.BICUBIC`

11) Что такое итератор и генератор? [1]

Simply speaking, a generator is a function that returns an object (iterator) which we can iterate over (one value at a time).

Iterator: An iterator is an object which contains a countable number of values and it is used to iterate over iterable objects like list, tuples, sets, etc.

Generator: It is another way of creating iterators in a simple way where it uses the keyword “yield” instead of returning it in a defined function. Generators are implemented using a function. Just as iterators, generators also follow lazy evaluation. Here, the yield function returns the data without affecting or exiting the function.

12) MapInPandas как использовать и для чего?

Maps an iterator of batches in the current `DataFrame` using a Python native function that takes and outputs a pandas DataFrame, and returns the result as a `DataFrame`.

```
df.mapInPandas(filter_func, df.schema)
```

13) Как задается Schema в spark?

#checkup_7 вопрос 3.

14) За что отвечает параметр mode в обработке изображений в спарк?

Численное представление поддерживаемых библиотекой OpenCV (#checkup_8 вопрос 9).

15) Как поменять размер картинки?

Выбрать конечный размер, способ интерполяции пикселей (заполнения пустот при сужении/расширении):

```
img = img.resize([224, 224], resample=Image.Resampling.BICUBIC)
```

16) Как получить топ-5 предсказаний из **resnet50**?

Подаём список предсказаний и выбираем топ-X (первых) нужных.

```
def top5_predictions(preds):
    return tf.keras.applications.resnet50.decode_predictions(
        np.array(preds).reshape(1,1000), top=5
    )
```