

# Big Data


На примере Small Data






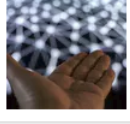
# Big Data (определение из Wikipedia)









- **Volume:** big data doesn't sample; it just observes and tracks what happens
- **Velocity:** big data is often available in real-time
- **Variety:** big data draws from text, images, audio, video; plus it completes missing pieces through data fusion
- ***Machine Learning*:** big data often doesn't ask why and simply detects patterns
- ***Digital footprint*:** big data is often a cost-free byproduct of digital interaction

# Big Data на Coursera

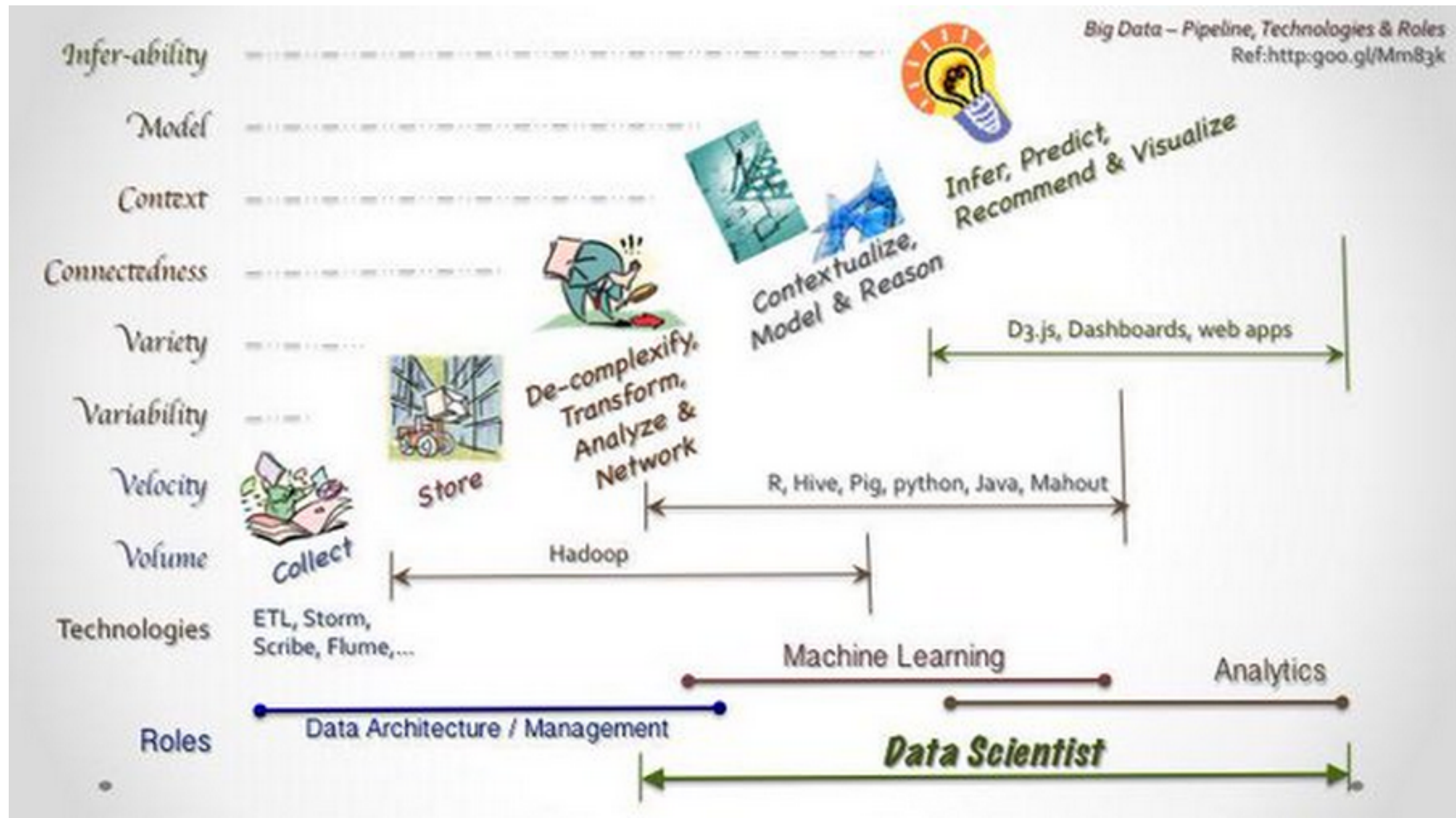
**Машинное обучение ?**

Активные фильтры: 

-  **Машинное обучение**  
Специализация из 6 курсов · Вашингтонский университет
-  **Машинное обучение и анализ данных**  
Специализация из 6 курсов · Московский физико-технический институт
-  **Big Data**  
Специализация из 6 курсов · Калифорнийский университет в Сан-Диего
-  **Probabilistic Graphical Models**  
Специализация из 3 курсов · Стэнфордский университет
-  **Robotics**  
Специализация из 6 курсов · Пенсильванский университет
-  **Recommender Systems**  
Специализация из 5 курсов · Миннесотский университет

-  **Наука о геномных данных**  
Специализация из 8 курсов · Университет Джона Хопкинса
-  **Applied Data Science with Python**  
Специализация из 5 курсов · Мичиганский университет
-  **Big Data**  
Специализация из 6 курсов · Калифорнийский университет в Сан-Диего
-  **Data Analytics for Business Bootcamp**  
Специализация из 5 курсов · Колорадский университет в Боулдере
-  **Структуры данных и алгоритмы**  
Специализация из 6 курсов · Калифорнийский университет в Сан-Диего, Высшая школа экономики
-  **Наука о больших данных**  
Специализация из 4 курсов · Вашингтонский университет
-  **Data Analysis and Presentation Skills: the PwC Approach**  
Специализация из 5 курсов · PwC
-  **Análisis de Datos para la toma de decisiones**  
Специализация из 5 курсов · Технологический университет Монтеррея

# Этапы, технологии, роли



# Spark

## spark-notebook.io

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.mllib.rdd.RDDFunctions._
import scala.util.{Try, Success, Failure}
import java.sql.Timestamp

val sqlContext = new SQLContext(sparkContext)
import sqlContext.implicits._
```

# Идеи по BTCUSD на TradingView

4000 опубликованных идей получены примерно так:

```
curl -s "https://www.tradingview.com/chart/?stream=bitcoin&time=all&s=0&l=1000"
```

```
curl -s "https://www.tradingview.com/chart/?stream=bitcoin&time=all&s=1000&l=1000"
```

```
curl -s "https://www.tradingview.com/chart/?stream=bitcoin&time=all&s=2000&l=1000"
```

```
curl -s "https://www.tradingview.com/chart/?stream=bitcoin&time=all&s=3000&l=1000"
```

## Распарсим время

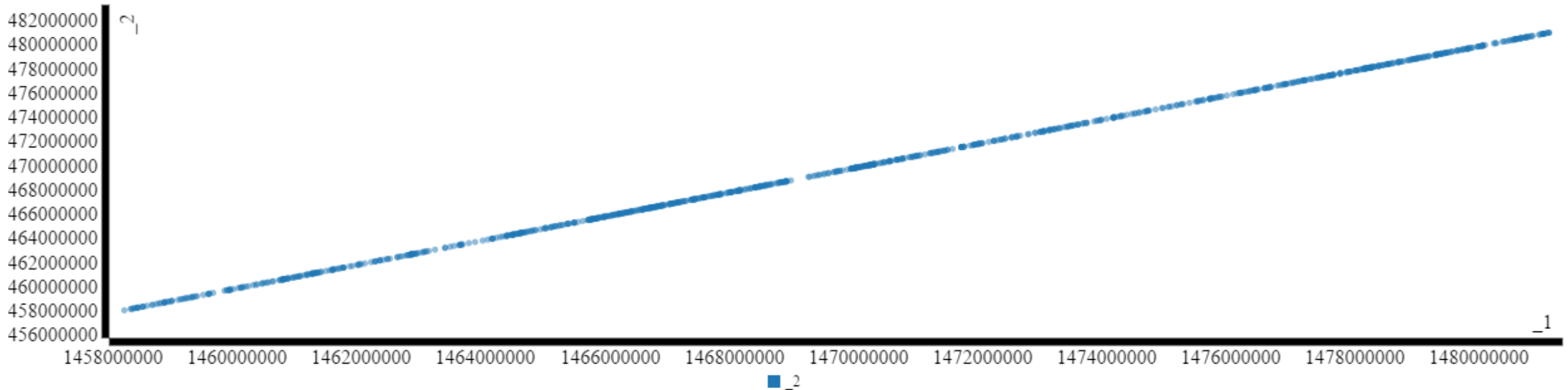
```
val btcIdeas = sparkContext.textFile("data/1000_1_BTC_ideas.json")  
    .union(sparkContext.textFile("data/1000_2_BTC_ideas.json"))  
    .union(sparkContext.textFile("data/1000_3_BTC_ideas.json"))  
    .union(sparkContext.textFile("data/1000_4_BTC_ideas.json"))
```

```
import java.lang.Double  
val times = btcIdeas  
    .flatMap(_.split("data-timestamp"))  
    .map(_.split("\\\\")(1))  
    .map(x => x.substring(0, x.length - 1))  
    .flatMap(x => Try(Double.parseDouble(x)).toOption)
```

```
times: org.apache.spark.rdd.RDD[Double] = MapPartitionsRDD[14] at flatMap at :73
```

# Идеи по BTCUSD на шкале времени

```
ScatterChart(times.map(x => (x,x)).collect)
```



## Время первой и последней собранной идеи

```
val t = Seq(times.collect.min, times.collect.max)
    .map(x => new Timestamp(x.toLong * 1000));
```

```
t: Seq[java.sql.Timestamp] = List(2014-07-28 19:30:18.0, 2016-12-07 19:59:13.0)
```



# Новости Yahoo

В `data/yahoo.txt` - манипуляциями с html в текстовом редакторе собраны новости по BTCUSD в виде списка ссылок:

```
http://finance.yahoo.com/news/exclusive-mona-el-isa-google-171602365.html  
http://finance.yahoo.com/news/exclusive-mona-el-isa-google-161420691.html  
http://finance.yahoo.com/news/payments-marijuana-industry-blockchain-increase-163357208.  
http://finance.yahoo.com/news/youre-money-ny-judge-rules-162025300.html  
...
```

```
val btcNewsUrls = sparkContext.textFile("data/yahoo.txt")
```

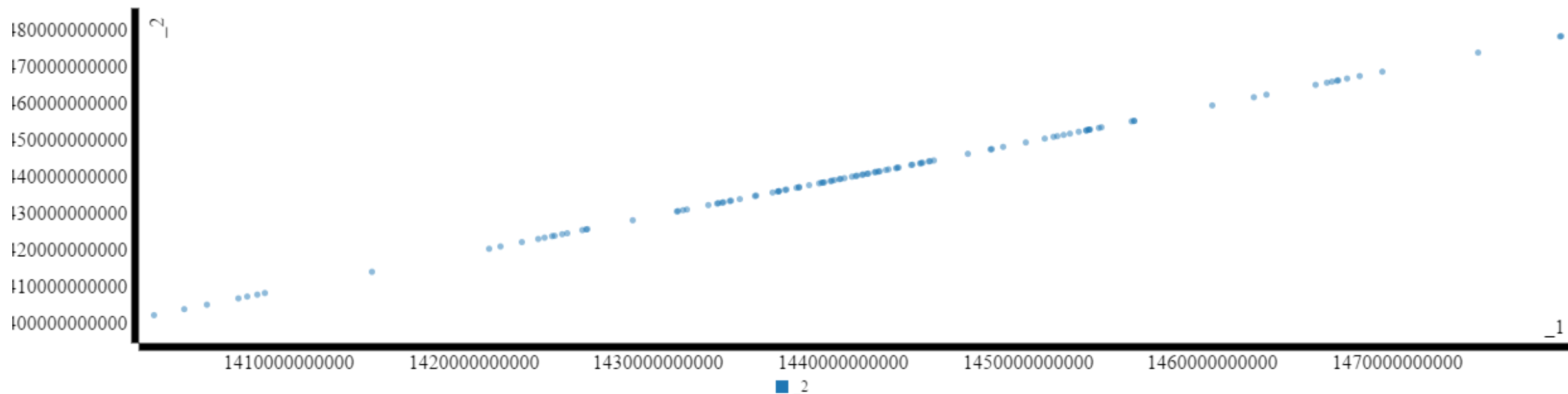
## Соберём и распарсим время новостей

```
import java.text.SimpleDateFormat
val dateFormatter =
    new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'")
val yahooTimes = btcNewsUrls.flatMap(x => Try{
    val s = scala.io.Source.fromURL(x, "utf-8")
    val res = s.getLines
        .filter(_.contains("datetime"))
        .take(1)
        .toList

    s.close()
    res
}.toOption)
.flatMap(x => x)
.flatMap(x => "datetime=\"(.*)\"".r.findFirstMatchIn(x).map(_.group(1)))
.map(dateFormatter.parse(_))
.map(_.getTime)
.collect
```

# Yahoo новости по BTCUSD на шкале времени

```
ScatterChart(yahooTimes.map(x => (x,x)))
```



## Время первой и последней собранной новости

```
val t = Seq(yahooTimes.min, yahooTimes.max)  
  .map(x => new Timestamp(x.toLong));
```

```
t: Seq[java.sql.Timestamp] = List(2014-06-13 18:02:44.0, 2016-11-11 17:16:02.0)
```

# Сам BTCUSD в это время

apozdnyakov published on TradingView.com, December 25, 2016 23:09 MSK

BITSTAMP:BTCUSD, D 887.14 ▼-5.47 (-0.61%) O:892.61 H:892.61 L:859.00 C: 887.14



# Сравним идеи и новости

Сперва приведём к одному формату

```
val newsTimes = yahooTimes.sorted  
val ideasTimes = times.map(_ * 1e3).collect.sorted
```

# Статистика

Сравнивать точки во времени не совсем понятно как.

Будем сравнивать плотности вероятностей.

```
import org.apache.spark.mllib.stat.KernelDensity
import org.apache.spark.rdd.RDD

def myKernelDensity(arr: Array[Double]): KernelDensity = {
  val data = sc.parallelize(arr)
  new KernelDensity().setSample(data).setBandwidth(3e8)
}

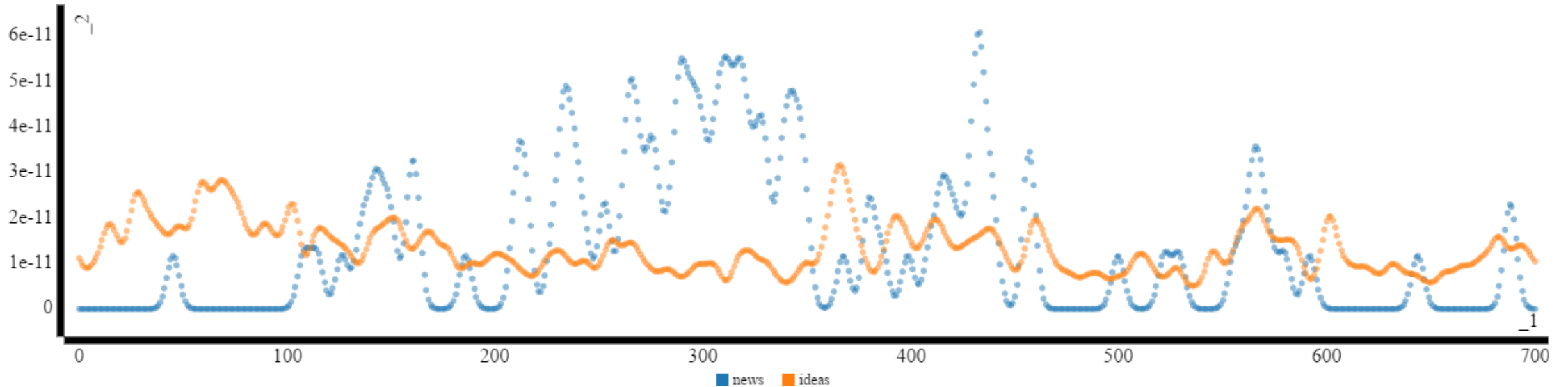
val kdi = myKernelDensity(ideasTimes)
val kdn = myKernelDensity(newsTimes)

// 14100 to 14800 * 1e8 - it is time: 06 Sep 2014 - 24 Nov 2016
val densitiesi = kdi.estimate((14100 to 14800).map(_ * 1e8).toArray)
val densitiesn = kdn.estimate((14100 to 14800).map(_ * 1e8).toArray)
```

# На графике

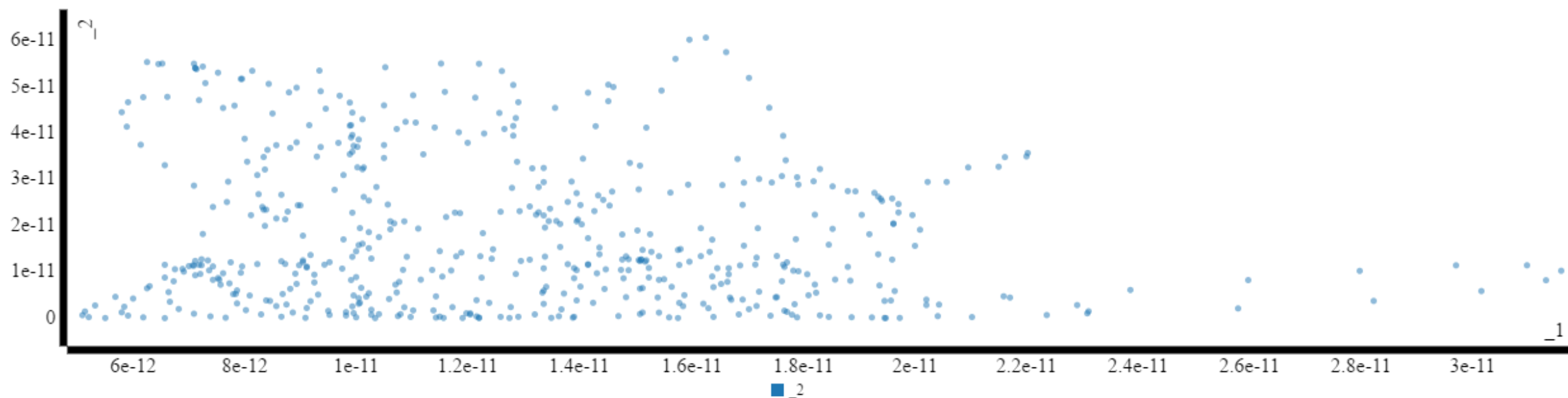
```
val newsAndIdeasForChart = (densitiesn zip densitiesi).zipWithIndex
    .flatMap{ case ((x,y), i) => Seq((i, x, "news"), (i, y, "ideas")) }
```

```
ScatterChart(newsAndIdeasForChart,
    fields=Some("_1", "_2"),
    groupField=Some("_3"),
    maxPoints=2000)
```



## В другом виде

```
ScatterChart((densitiesi zip densitiesn).filter(_._2 >= 1e-13))
```





# Корреляция

```
import org.apache.spark.mllib.linalg._  
import org.apache.spark.mllib.stat.Statistics
```

```
val correlation: Double = Statistics.corr(  
  sc.parallelize(densitiesi),  
  sc.parallelize(densitiesn),  
  "spearman"  
)
```

```
correlation: Double = -0.10929748013913299
```

# Отбрасывая историю

Отбросим начало истории, 2/7

```
val correlation: Double = Statistics.corr(  
  sc.parallelize(densitiesi drop 200),  
  sc.parallelize(densitiesn drop 200),  
  "spearman"  
)
```

```
correlation: Double = 0.12755330772717519
```

Уже лучше. Отбросим ещё 2/7

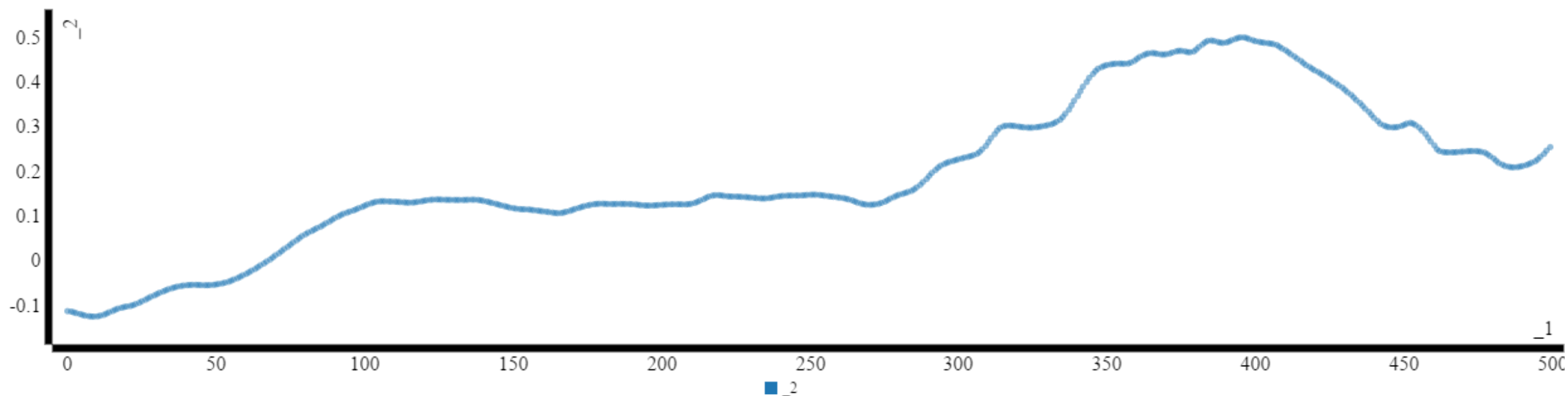
```
val correlation: Double = Statistics.corr(  
  sc.parallelize(densitiesi drop 400),  
  sc.parallelize(densitiesn drop 400),  
  "spearman"  
)
```

```
correlation: Double = 0.4965642120085375
```

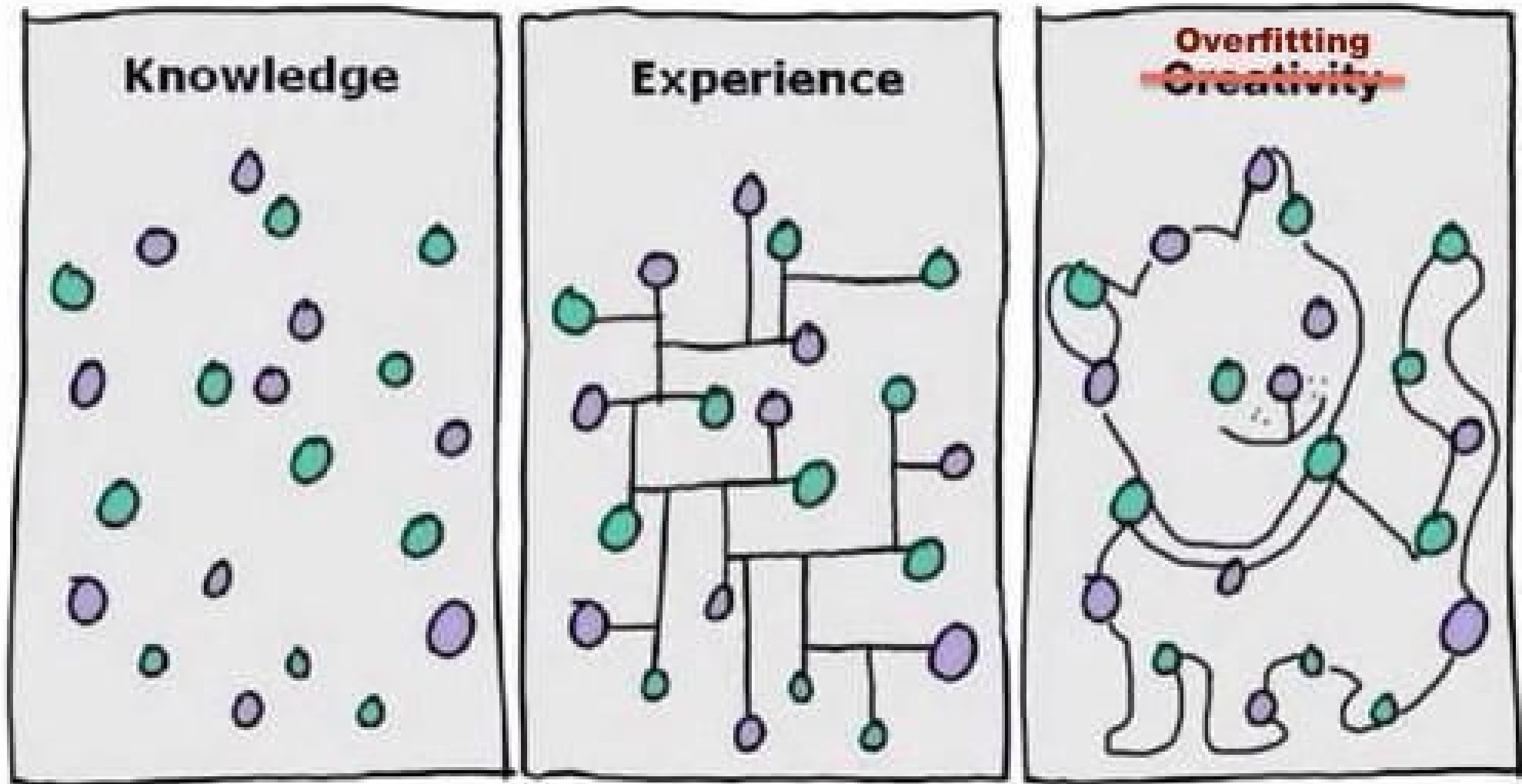
## Отбрасывая историю (2)

```
val correlations = for (i <- 1 to 500) yield Statistics.corr(  
  sc.parallelize(densitiesi drop i),  
  sc.parallelize(densitiesn drop i),  
  "spearman"  
)
```

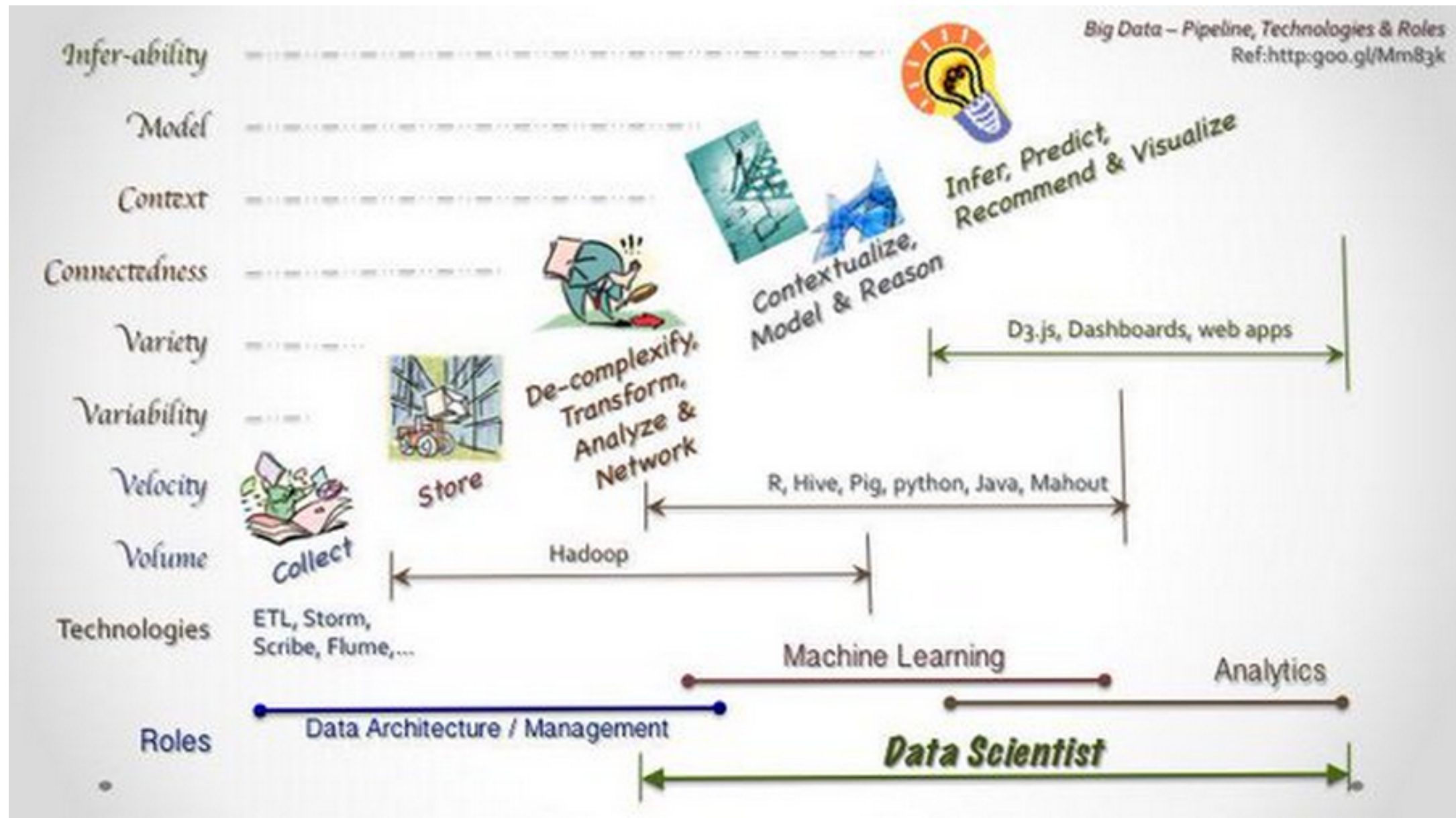
**ScatterChart**(correlations.toArray)



# Overfitting



## Этапы, технологии, роли (2)



## Для чего можно это использовать

- Публиковать блог-посты с анализом социальной активности.
- Рекомендовать пользователей, на которых надо подписаться
- Рекомендовать похожие опубликованные идеи.
- Сделать социальный датафид на наших чартах, выводящий KernelDensity новостей.
- Построить супер-мега-выигрышную стратегию на основе собранных знаний пользователей.