# 执行力

# 如何与客户沟通

# 书到用时方恨少

学的时候总抱怨方法、原理太多，真正用的时候发现一个也用不上。学的时候总感觉能做好大的项目，用的时候一个特征汇总就搞掉你。

介绍一些架构。

# 不平衡分类

# 有限总体稀有事件抽样

# 做事要做漂亮

在规定时间内做事，宁愿做的不要那么多也要那么精。

# 数据归一化对算法的影响

1. 一些算法天然的要求数据量纲一致：kmeans、knn、pca等
2. 除此之外，数据归一化还能够影响梯度下降等最优化方法的收敛。如神经网络在进行训练之前经常需要对量纲差异较大的数据进行归一化。

# 做一些项目的基本点

1）产品设计

对分布式来说，产品设计往往是其短板，由于其数据处理过程都是基于rdd的因此，在中间交互、结果展示等方面都受限

2）数据处理

3）展示

# 对于不定参数的对象

在应用中有些参数需要不定长的参数,比如有的需要列名/时间/分箱,有的需要列名/时间就可以.

此时两种可以解决上述需求:

1)将类写为class A(private var 参数1: T){以set形式传递参数}的形式

2)类正常写为class A(val 参数1: Option[T]）,以Option包括参数，直接apply

# 常用的工程项目目录结构和理念

## 理念

低内聚高耦合

## 目录方式1

Utils

Test

Models

Objects

# 为什么很多团队都强调结果导向

具备较强沟通能力，善于团队合作，交付能力强，自我驱动，结果导向。

不要过程导向或情感导向，只有结果很差，没有功劳时才谈苦劳。

1. 注重最终效果
2. 考量的内容首先是需求，其次才是实现。

# 产品设计

在设计的基本想法有了的时候，可以尝试用简短的几句话看看能不能把框架说清楚，如果不行再改一下。

# 对于只有很少数目范围的参数可以放在某个Object类中声明

以case class(fieldFormat: String, …)

fieldFormat可以为string、long等类型。

1. 一种方法是把“string”、“long”做成独立的类型，共同继承一个父类型。
2. 另一种是在case class实例化之前加入判定条件，fieldFormat必须在某个范围（“string”、“long”）内
3. 可以做另一个object

Object A extends Serializable{

Val firstType = “string”

Val secondType = “long”

}

Case class实例化时注意只输入A中的某个变量

# 需求管理

用的需求 > 开发需求 > 设计需求

设计时考虑的东西是少的，随着开发过程不断的思考，不断改正增加需求，在用的过程中又不断更新开发阶段的一些bug并提出一些新的设计

# 专业性

到一个地方应该注重显示自己的专业性，规范性，有其他人不合规范，即使略有棱角也要显示出来。虽然一开始可能还有点龃龉但后面他很快会正视你。

# 框架设计的一些注意事项

1. 里层的不能改变外层的变量

否则乱了。外层稍微一边动，里层的某个就不能运行了。牵一发动全身。最主要的是有时候忘记里层的某个组件改动外层的东西了，没有发现bug就是隐患，发现了bug也不好排查。

1. 高内聚，低耦合

目的也是和1一致，防止某个模块的稍一变动会导致其他模块跟着大的改动。

模块内可以大改，高内聚以充分利用代码和功能，避免一个脚本一个功能，而是要做成组合式的。

1. 模块划分要恰当

不要过大，会导致大的范围内内聚，牵一发动全身。

不要过小，会导致模块过多，耦合过多。