第一章 Python和数据化运营

企业的数据化运营是提高利润、降低成本、优化运营效率、最大化企业财务汇报的必要课题。

Python作为数据科学界的关键工具之一，几乎可以应用于所有数据化运营分析和实践的场景。

1.1 用 Python 做数据化运营

1.1.1 Python是什么？

Python是一种**面向对象**的**解释型**计算机程序设计语言，有荷兰人 Guido van Rossum 于1989年发明，1991年发行。

选择Python的原因：

- **开源/免费**：无需购买产品、授权、license
- **可移植性**：跨平台
- **丰富的结构化和非结构化数据工作库和工具**：除了自身的工具库之外，有着丰富的第三方库和工具，涵盖数据库、科学计算、文本处理、机器学习、图形视频分析处理等等
- **强大的数据获取和集成能力**：支持多种文件类型，可通过多种方式（API、抓取等）获取外部数据。内外数据整合、多源数据集成、异构数据并存、多类型数据交错正式当前企业数据运营的基本形态。
- **海量数据的计算能力和效率**：能够支撑GB甚至TB规模的海量数据
- **与其他语言集成**：Python具有“胶水”能力，能与多种其他语言集成使用
- **强大的学习交流和培训资源**：社区、博客、论坛、培训机构、教育机构等
- **开发效率高**：语言简洁、规范
- **简单易学**：语法简单

1.1.2 数据化运营是什么？

数据化运营是指**通过数据化的工具、技术和方法，对运营过程中的各个环节进行科学分析、引导和应用，从而达到优化运营效果和效率、降低成本、提高效益的目的。**

1. 数据化运营的意义

数据化运营的核心是**运营**，所有的数据工作都是围绕运营工作链条展开的，逐步强化数据对于运营工作的驱动作用。数据化运营的价值体现在对运营的辅助、提升和优化上，甚至某些运营工作逐步实现数字化、自动化、智能化。

- **提高运营决策效率**：短时间内做出决策，领先竞争对手。属于辅助决策方式。
- **提高运营决策正确性**：演练并得出可量化的预期结果，配合决策层的经验，提高决策的正确性。
- **优化运营执行过程**：为运营提供标准的、统一的数据格式、信息场景和结论依据等，优化运营中的执行环节
- **提升投资回报**：通过对持续的正确工作目标的确立、最大化工作效率的提升、最优化工作方法的执行能够有效的降低企业冗余支出，提升单位成本的投资回报

2. 数据化运营的2种方式

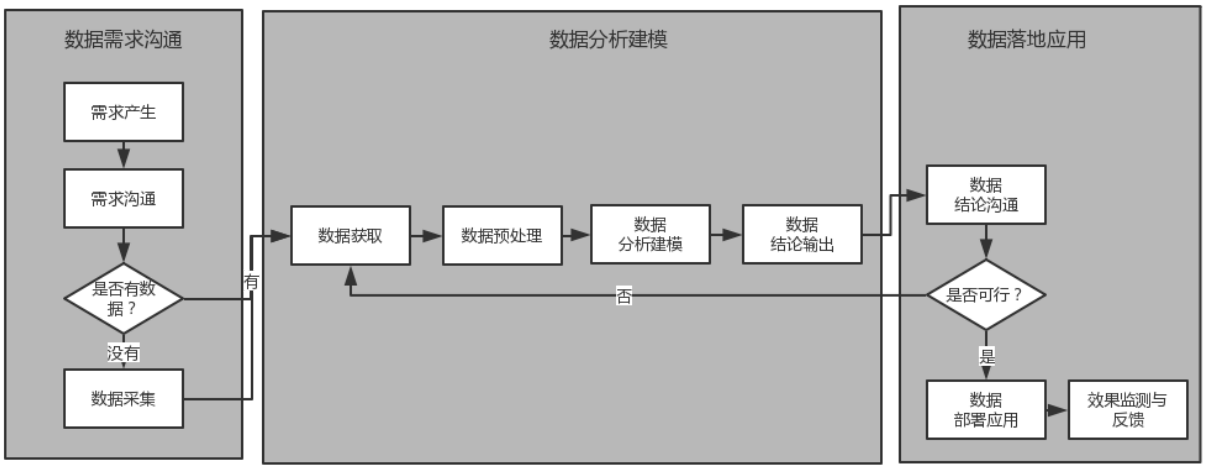
从数据发挥作用的角度来看，数据化运营分为**辅助决策式数据化运营**和**数据驱动式数据化运营**。

- **辅助决策式数据化运营**：是运营的决策支持，以决策主题为中心，借助计算机相关技术辅助决策者通过数据、模型、知识等进行业务决策，起到帮助、协助和辅助决策者的目的。为业务决策方服务，数据是辅助角色。

- **数据驱动式数据化运营**：是指整个运营运作流程以最大化结果为目标，以关键数据为触发和优化方式，将运营业务的工作流程、逻辑、技巧封装为特定应用，借助计算机技术并结合企业内部流程和机制形成一体化的数据化工作流程。数据是主体，具有自主导向性、自我驱动性和效果导向性。实现过程需要IT、自动化系统、算法等支持。

- 数据化运营的工作流程（数据驱动式数据化运营）

数据驱动式数据化运营工作包含**数据**和**运营**两个主体，实际工作中需要二者协同配合。



1.3 数据化运营所需的Python相关工具和组件

略

1.4 案例：第一个用 Python 实现的数据化运营分析实例 --- 销售预测

案例场景

每个销售型公司都有一定的促销费用，促销费用可以带来销售量的提升；当给出一定的销售费用时，预计会带来多大的商品销售量？

实例数据说明

- 来源：模拟数据，非真实数据
- 用途：用来做第一个销售预测案例
- 数据维度：1
- 样本数：100
- 个体变量：第一列为促销费用，第二列为商品销售量
- 数据类型：float
- 是否有缺失值：否

> 备注：样本数据为目录下**data.txt**文件。

案例过程

1. 导入必要的Python库
2. 导入数据
3. 数据预处理
4. 数据分析
5. 数据建模
6. 模型评估
7. 销售预测

1. 导入必要的Python库

在本案例中，会使用到如下的四个Python库：

- Re：正则表达式库
- Numpy：数组操作和计算库
- Sklearn：算法模型库
- Matplotlib：图形展示库

```
import re
import numpy as np
```

```
from sklearn import linear_model
import matplotlib.pyplot as plt
```

2. 导入数据

本案例使用的是txt格式的文本文件，因此直接使用Python默认的读取方式即可

```
# 仅用于切换工作目录（执行成功一次即可）
import os
os.chdir('C1 Python和数据化运营/')
```

```
# 读取方式一
file = open('data.txt', 'r')
all_datas = file.readlines()
file.close()
```

```
# 读取方式二
with open('data.txt', 'r') as fn:
    all_datas = fn.readlines()
fn.close()
```

```
all_datas[0]
```

```
'28192.0\t68980.0\n'
```

3. 数据预处理

对列表数据进行清洗切换，以满足数据分析展示和数据建模的需要

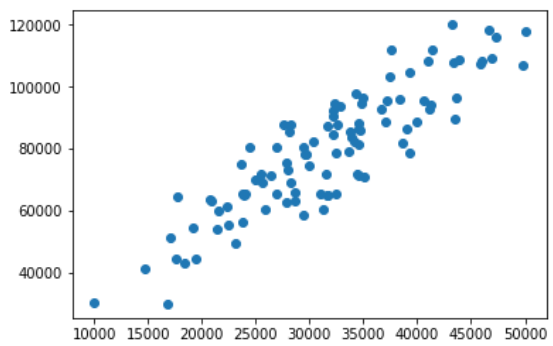
```
x = []
y = []

for single_data in all_datas:
    tmp_data = re.split('\t|\n', single_data)
    x.append(float(tmp_data[0]))
    y.append(float(tmp_data[1]))
```

```
x = np.array(x).reshape([100, 1])
y = np.array(y).reshape([100, 1])
```

4. 数据分析

格式化的数据已经准备好，但是改用哪种模型还未知，可以先将数据可视化，从可视化图像中寻找线索。

```
plt.scatter(x, y)
plt.show()
```



5. 数据建模

使用SKlearn中线性回归模块建模。

```
# 初始化算法模型
lr_model = linear_model.LinearRegression()
# 拟合数据
lr_model.fit(x, y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

6. 模型评估

模型创建完成，进行模型拟合的校验和评估。

对于线性回归算法来说，通常的评估是获取模型拟合后的 `coef` 和 `intercept` ，他们分别代表了线性回归方程中的**斜率**和**截距**。以及线性回归的R平方值。

```
model_coef = lr_model.coef_
model_coef
```

```
array([[ 2.09463661]])
```

```
model_intercept = lr_model.intercept_
model_intercept
```

```
array([ 13175.36904199])
```

由上述结果可得到线性回归方程为：

y = 2.09463661 * x + 13175.36904199

```
# R平方值
r2 = lr_model.score(x, y)
r2
```

```
0.78764146847589545
```

R平方值可以理解为模型的预测正确率，也就是78.8%。

7. 销售预测

假设有一项促销费用（x）为84610，预测其带来的商品销售量

```
# 手动计算
x = 84610
y = 2.09463661 * x + 13175.36904199
y = 190402.57234224601
```

```
# 模型预测
new_x = 84610
pred_y = lr_model.predict(new_x)
pred_y[0][0]
```

```
190402.57234224601
```

```
round(pred_y[0][0])
```

```
190403.0
```

总结

本部分内容并没有讲解过多的算法实现和参数说明，仅仅是为了能够对Python、数据化运营有一个直观的感受和了解。