

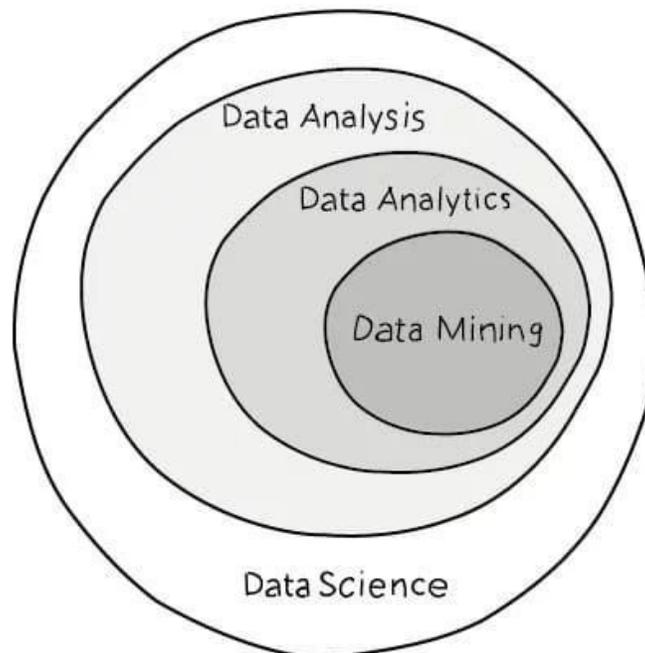
Introdução

Em um mundo cada vez mais conectado e tecnológico a produção de dados é exponencial.

Dispositivos diversos produzem dados de uma forma acelerada e trabalhar com um volume de dados em distintos formatos e estruturas é sem dúvida um grande desafio.

Podemos definir **Data Science** de uma forma bem simplificada e objetiva como uma habilidade para manipular, analisar e extrair valor dos dados. Valiosos insights podem ser gerados a partir da descoberta de conhecimento e inteligência adquirida.

Data Science combina métodos de diversas áreas como estatística, análise de dados, computação e suas tarefas relacionadas com o objetivo de olhar para os dados de uma maneira diferente para encontrar padrões, tendências, fazer previsões a fim de agregar valor para o negócio.



Porque Aprender Data Science?

Uma pesquisa realizada em 2012 pela [Gartner](#), concluiu que o Brasil precisará de um número imenso de profissionais para trabalhar com Data Science.

Este estudo ainda mostra que atualmente há um déficit de profissionais de TI no Brasil e que esse déficit se tornará ainda maior com a demanda de profissionais nessa área.

O Gartner prevê que os especialistas em dados serão muito valorizados no mercado mundial e que apenas $\frac{1}{3}$ das vagas seriam preenchidas em 2015. Veja esse trecho da pesquisa:

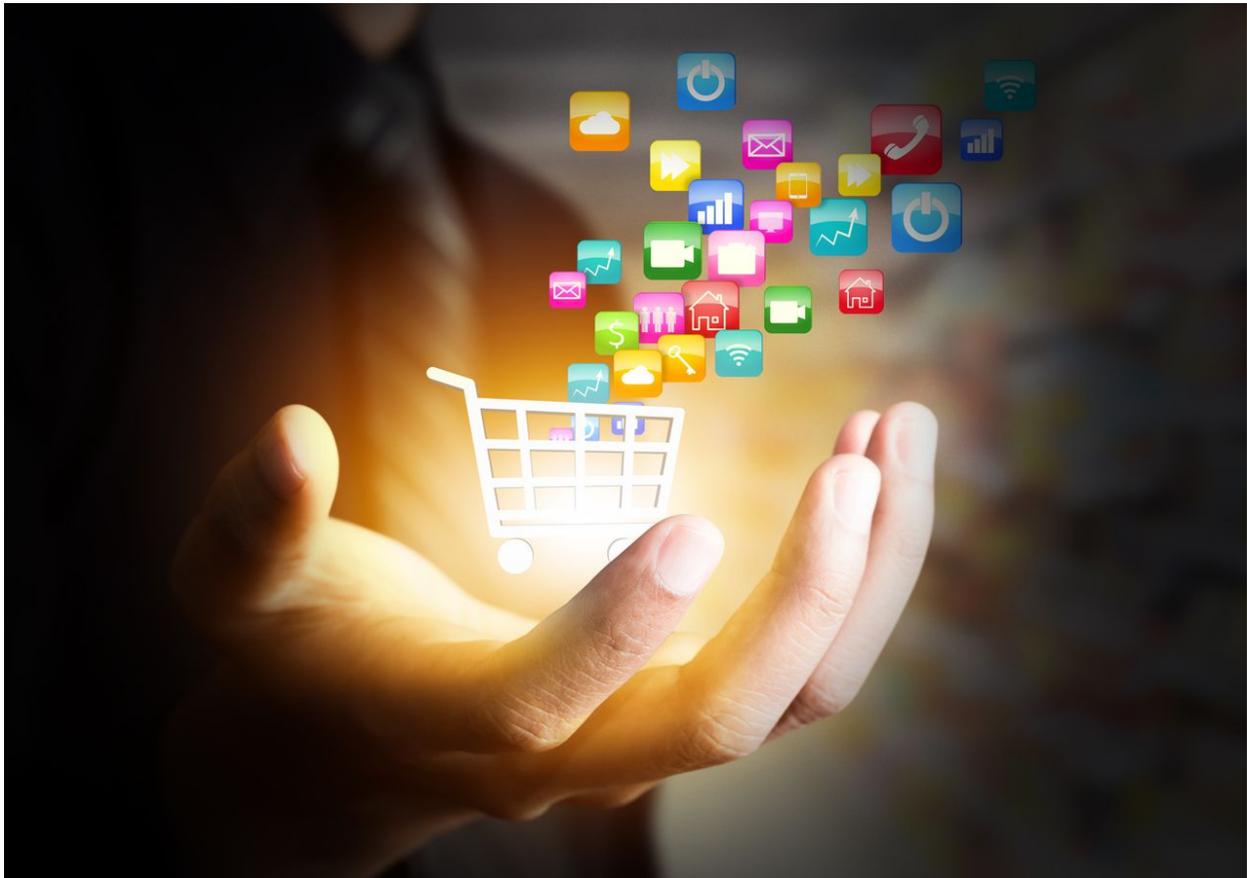
“Essa área vai movimentar a economia no mundo todo, mas apenas 1/3 dos cargos será preenchido”, afirmou Sondergaard durante o Gartner Symposium/ITxpo2012.

Data Science e algumas aplicações



Existem diversas situações onde podemos aplicar Data Science para resolver problemas importantes. Vejamos alguns cenários e suas aplicações.

Logística e Varejo



Predizer a quantidade de vendas de um produto ou serviço baseado em seus dados históricos, fatores climáticos, eventos externos e preferências do cliente pode ser interessante, certo?

Podemos usar Data Science para descobrir se a sazonalidade no número de vendas de algum produto está atrelada a algum fator que não esteja tão claro para a empresa, ou até mesmo, otimizar o estoque com números mais assertivos para a compra de produtos para a revenda.

Reduzir custos de entrega levando em consideração dados de geolocalização e ao mesmo tempo proporcionar uma melhor experiência de compra para o cliente pode ser um grande diferencial.

Tecnologia e Entretenimento



Recomendação de serviços e produtos baseado nas preferências do usuário pode ser interessante pela maior assertividade nas ofertas e por proporcionar uma experiência exclusiva.

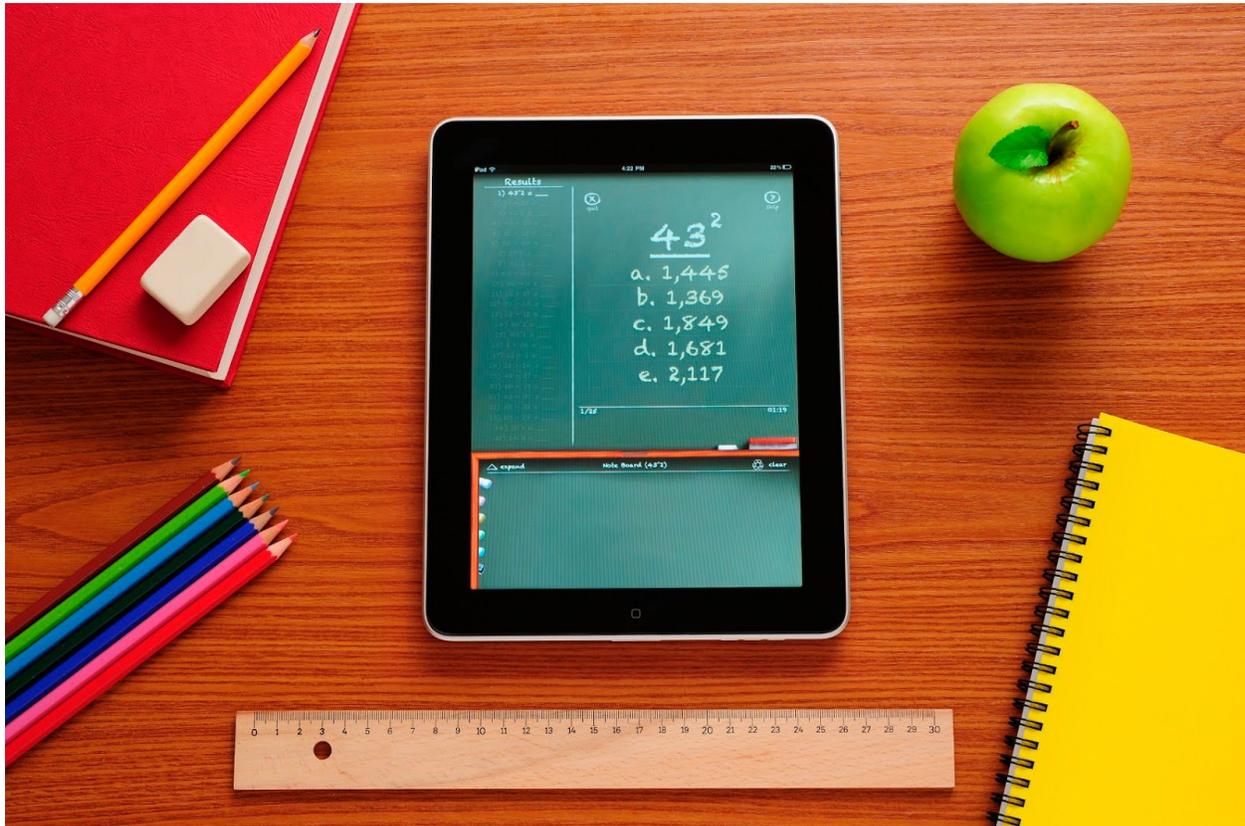
Essa é uma das apostas da [Netflix](#) que recomenda filmes e séries baseadas no histórico de acesso dos seus usuários. Utilizando o seu grande volume de dados e o padrão de consumo dos usuários o sistema consegue prever com altíssima precisão quais os filmes e séries o usuário pode se interessar.

Além de recomendar os filmes certos, a empresa utiliza esse conhecimento para produzir séries originais como é o caso de House of Cards.

Outra grande empresa que utiliza o poder do Data Science para vender mais e melhorar a experiência do cliente é a [Amazon](#) através de sistemas de recomendação baseado no histórico de compras e visitas dos usuários.



Educação



Modernizar o ensino não é mais uma tendência ou promessa para o futuro e sim uma necessidade. A forma como as pessoas aprendem já não é a mesma e a tecnologia é um fator imprescindível para viabilizar essa transformação.

Com o surgimento de novas ferramentas e canais de ensino tornou possível trabalhar dados que antes se tornavam inúteis e obsoletos.

Técnicas de Data Science podem ser aplicadas para medir o desempenho de alunos e professores permitindo um melhor aproveitamento dos recursos através da identificação de padrões de aprendizado.

Alunos com maiores dificuldades no aprendizado de determinadas disciplinas podem ser separados e encaminhados para um atendimento personalizado. Através da predição de desempenho a partir das avaliações e frequência dos alunos a instituição pode antecipar o acompanhamento e combater a evasão escolar.

Além de permitir uma maior experiência para os alunos a instituição pode aprender com o processo orientado a dados.

Saúde



Com tecnologias cada vez mais poderosas que coletam e produzem dados vitais para o monitoramento e acompanhamento em tempo real do corpo humano Data Science surge como uma ferramenta fundamental para ajudar em um diagnóstico eficiente e inteligente baseado em dados.

Sistemas Cognitivos estão sendo desenvolvidos com a premissa de prever o estado de saúde de pacientes de alto risco, baseado em dados de clientes empresas de planos de saúde tem utilizado Data Science para oferecer serviços médicos cada vez mais personalizados.

Com o crescimento de pequenos dispositivos de monitoramento de sinais vitais acoplados em celulares, relógios, médicos e instituições de saúde podem acessar e analisar dados de pacientes e oferecer um tratamento de forma preventiva.

Data Science aplicado a saúde além de possibilitar uma gestão eficiente de recursos, possibilita uma transformação inteligente de como atores se interagem para prevenir doenças e melhorar a qualidade de vida das pessoas.

Empresas que usam Data Science em seus negócios

Atualmente existem diversas Empresas\Startups no Brasil que já usam Data Science em seus negócios. Esses são negócios inovadores que já demandam de uma mão de obra especializada nessa área. Veja alguns exemplos:

- **Vérios:** A Véríos é considerada uma fintech por ter seu negócio na área de investimentos. Inovadora, a Véríos possui um serviço chamado “Carteira inteligente”. A carteira inteligente da Véríos usa um Robot que faz aplicações de forma automática sempre buscando as melhores oportunidades do mercado financeiro.
- **Nuveo:** A Nuveo é uma empresa que possui diversos produtos para automatização de processos de seus clientes. Esta faz coleta automática de dados, processamento e inteligência usando uma plataforma de sistemas inteligentes. Um dos produtos da Nuveo, é o *Contract Intelligence*. Este produto faz a descoberta de informações relevantes de contratos. O CI interpreta contratos dos mais diversos tipos usando processamento de linguagem natural e redes neurais.
- **Hekima:** A Hekima desenvolve e aplica tecnologias de Data Science, com o objetivo de ajudar empresas a transformarem dados em informação para tomada de decisão e, assim, melhorarem a performance de negócios. A Hekima entrega soluções de inteligência artificial como serviços para seus clientes.
- **goGeo:** A goGeo é uma Startup que contém uma plataforma que faz Data Science de dados geográficos. Com centenas de milhares de dados, esta faz o processamento e gera valiosos insights sobre redes sociais, segmentos de mercado, entre outros. Além do mais, a goGeo disponibiliza Api's para desenvolvedores integrarem suas aplicações com os dados deles.
- **BigData Corp:** A BigData Corp tem como objetivo ajudar empresas a utilizar o potencial do Big Data em seus negócios. A empresa conta com produtos que podem ser usados para gerar leads, segmentar mercados ou ainda automatizar processos. Sem dúvidas essa empresa é uma das mais inovadoras que utilizam Big Data no Brasil.
- **ShopBack:** A Shopback usa Data Science para ajudar e-commerces a aumentarem suas conversões. Esta empresa usa tecnologia avançada para monitorar o comportamento de usuários, identificar novos usuários evitando que este abandone o

site. Seus sistemas inteligentes ainda exibem diversas estatísticas para que o cliente possa usar esse conhecimento para aumentar suas conversões.

- **Tracksale:** A Tracksale faz uma análise da satisfação do cliente. Um dos seus produtos é a classificação automática de comentários de clientes em Positivo, Negativo ou Neutro. Essa tarefa também conhecida como Análise de Sentimentos, é uma das funcionalidades de sucesso que usam técnicas de Data Science da Tracksale.

Ferramentas Essenciais para Trabalhar com Data Science.

Neste capítulo vamos conhecer e instalar as ferramentas, linguagens e bibliotecas **essenciais** para trabalhar com Data Science.

Produtividade é um fator decisivo para o sucesso de qualquer projeto e se tratando de Data Science não seria diferente. Saber utilizar as ferramentas certas para cada tarefa não só resultará em um trabalho com mais qualidade como também evita erros e reduz esforços desnecessários.

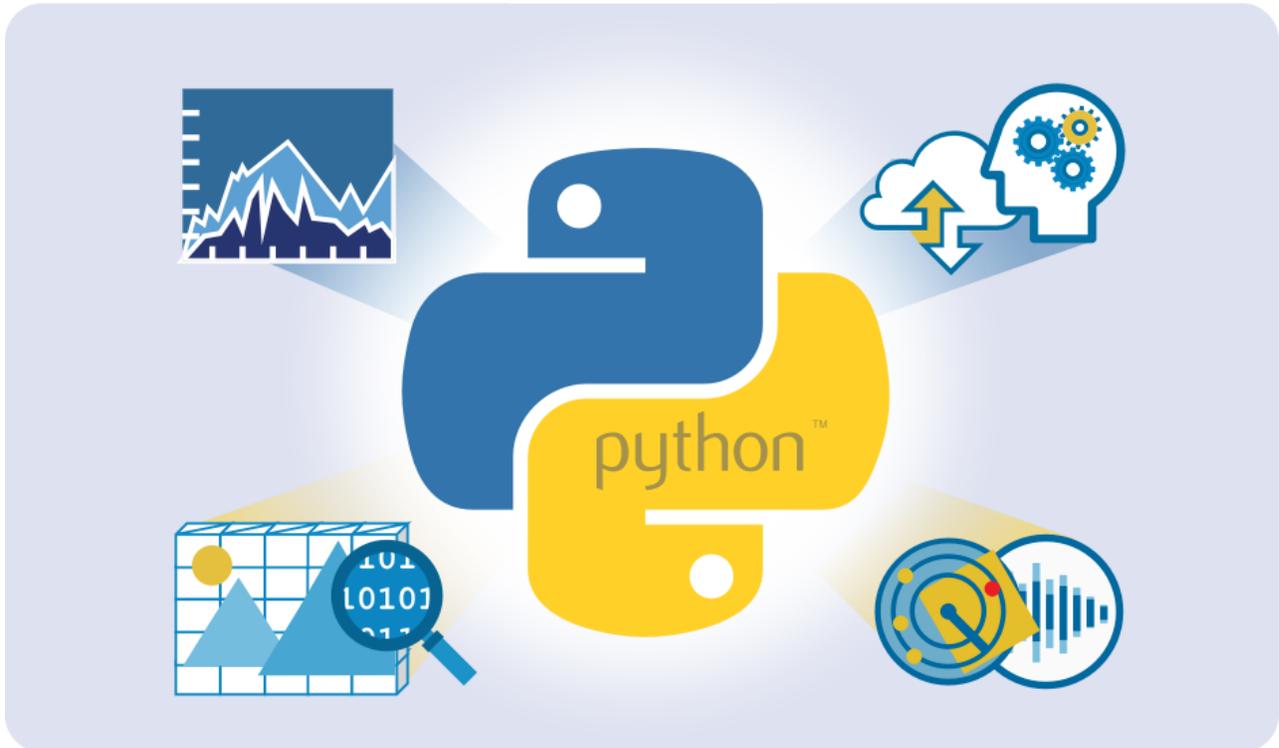
Diante do desafio de extrair valor dos dados temos diversas opções com os mais variados objetivos e cabe a você Cientista de Dados tomar a decisão sobre qual ferramenta ou linguagem utilizar para resolver o problema, levando em consideração variáveis como custo, complexidade, escalabilidade dentre outras.

Veremos as ferramentas mais utilizadas pela comunidade Data Science e quando utilizar cada uma, avaliando suas vantagens e desvantagens e como podemos explorar o melhor em cada situação.

Durante a fase de coleta e preparação dos dados, por exemplo, pode ser mais interessante utilizar uma linguagem de programação devido a sua flexibilidade e desempenho. Já para visualizar e interagir com os dados já tratados uma ferramenta de visualização seja mais apropriada pelas suas características para exibir a informação. Para armazenar e consultar os dados trabalhar com um banco de dados pode trazer diversos benefícios como segurança, escalabilidade e performance.

Como podemos ver, diferentes fases do projeto irão exigir diversas habilidades e além de avaliar as melhores ferramentas levando em consideração os fatores que destacamos acima é de extrema importância o conhecimento do negócio e seus recursos disponíveis.

Python: Agora ninguém poderá te deter!



Uma [pesquisa](#) em maio de 2015 revelou as ferramentas mais usadas pela comunidade de Data Science do site **KDnuggets**. Segundo a pesquisa, **Python** foi considerada a linguagem de programação mais usada pela comunidade de cientistas de dados.

Python tem inúmeras razões para ser escolhida e podemos citar algumas como **simplicidade**, **clareza** e **reusabilidade**. A linguagem oferece uma sintaxe simples e objetiva permitindo o programador se focar no problema a ser resolvido sem se preocupar tanto com detalhes de implementações.

Por exigir que o código fonte seja corretamente endentado, sua leitura e compreensão se torna extremamente clara e organizada, contribuindo para o aumento de produtividade entre programadores.

Além da grande comunidade no mundo inteiro, Python possui um vasto e variado conjunto de bibliotecas para se trabalhar com diversas áreas, desde computação científica, redes, segurança e claro análise de dados.

Vamos citar algumas principais ferramentas e bibliotecas indispensáveis para trabalhar com Data Science.



Jupyter: Aplicação cliente-servidor que permite a edição e execução de notebooks via browser. Notebooks são documentos que contêm código e elementos visuais como imagens, links, equações.

A principal vantagem na utilização de notebooks é para a documentação de análises e seus resultados de forma dinâmica e interativa além de permitir o rápido compartilhamento através da sua arquitetura web.

O Jupyter utiliza o conceito de Kernels, onde é possível utilizar diferentes linguagens de programação para executar scripts no mesmo Notebook. Para documentação por padrão é disponibilizado a linguagem Mark Down que é amplamente utilizada e compatível com os principais produtos como navegadores etc.

O projeto Jupyter é muito utilizado pela comunidade de Data Science ao redor do mundo.



ANACONDA[®]

Powered by Continuum Analytics

Diante da frustrante tarefa de manter o ambiente de pacotes para trabalhar com Data Science homogêneo o [Anaconda](#) vem para resolver esse problema fornecendo um ambiente open source e totalmente integrado com centenas de pacotes para trabalhar com ciência, matemática, engenharia e análise de dados.

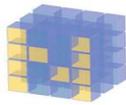
Além dos mais populares e confiáveis pacotes já estarem presentes no pacote do anaconda é fornecido um aplicativo para o gerenciamento de novos pacotes chamado [conda](#). Através do conda é possível remover pacotes e bibliotecas indesejáveis e adicionar outras em seu ambiente caso seja necessário.

O Anaconda trabalha com Python e R e tem versões para executar tanto em ambiente Linux, Windows e OSX.

Veremos mais adiante como instalar o configurar o Anaconda para trabalhar com Python e suas bibliotecas.

Bibliotecas Python

O Python como já descrevemos anteriormente é uma linguagem de programação bem completa e possui inúmeras bibliotecas para trabalhar com Análise de Dados. Veremos abaixo uma breve descrição das bibliotecas mais utilizadas e suas aplicações.



NumPy

NumPy : Biblioteca Python para computação científica. Implementa arrays multidimensionais e permite a fácil execução de operações matemáticas e lógicas como ordenação, seleção, transformações, operações estatísticas básicas etc.



tudo de forma fácil e rápida.

Matplotlib : Biblioteca Python 2D para a visualização e plotagem de gráficos. Pode ser utilizada para gerar diversos tipos de gráficos como histogramas, gráficos de barras, gráficos de pizza



$y_{it} = \beta x_{it} + \mu_i + \epsilon_{it}$

Pandas : Esta biblioteca talvez seja a mais utilizada para análise de dados. Ela fornece ferramentas para manipulação de estruturas de dados de forma extremamente simples. Operações complexas que trabalham com matrizes e vetores podem ser facilmente realizadas com uma ótima performance.

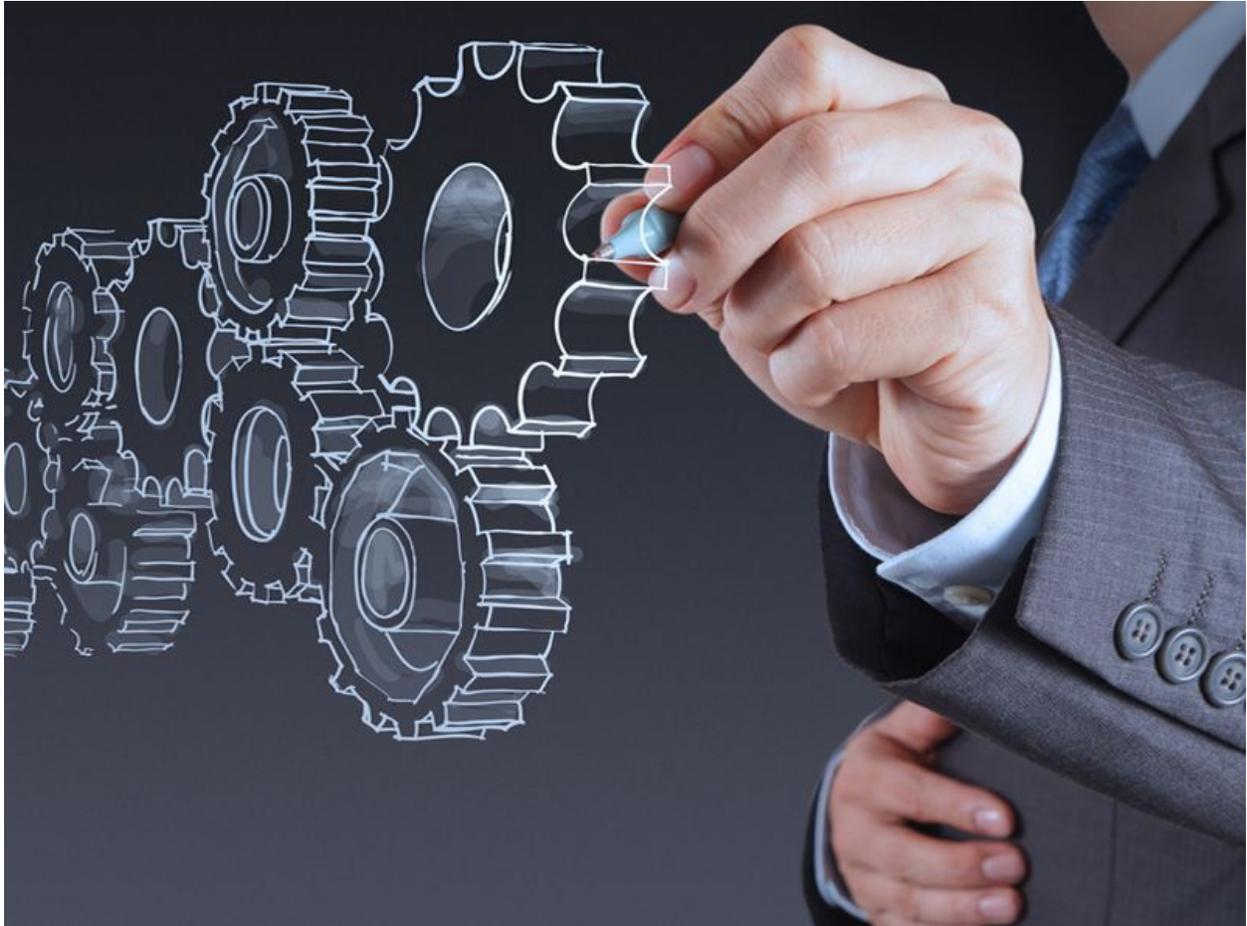


Scikit-Learn : Biblioteca Python para trabalhar com Machine Learn (Aprendizado de Máquina). Contém diversos algoritmos implementados, métodos de análise e processamento de dados, métricas de avaliação etc. Essa é uma biblioteca extremamente útil para o cientista de dados.

NLTK

NLTK: NLTK é uma plataforma líder para a construção de programas Python para trabalhar com dados de linguagem humana. Ele fornece interfaces fáceis de usar para mais de 50 corpora e recursos lexicais como o WordNet, juntamente com um conjunto de bibliotecas de processamento de texto para classificação, tokenização, stemming, tagging, análise e raciocínio semântico.

Instalação e Configuração do Ambiente



Neste capítulo iremos instalar o Anaconda e verificar as bibliotecas que já estão disponíveis utilizando o Python.

Para fazer download do Anaconda, clique [aqui](#)

A instalação é bem simples, veja o passo a passo para instalar a plataforma em ambientes Linux:

Abra o terminal e execute o instalador com comando abaixo:

```
bash Anaconda3-4.3.0-Linux-x86_64.sh
```

Após a instalação confira se a instalação foi bem sucedida conforme o comando abaixo:

```
~/anaconda3/bin$ ./python --version  
Python 3.6.0 :: Anaconda 4.3.0 (64-bit)
```

As principais bibliotecas que abordamos já estão instaladas, veja:

```
~/anaconda3/bin$ ./python  
import pandas as pd  
import numpy as np  
import matplotlib as plt  
from sklearn import datasets
```

Falta apenas instalar o Scrapy e o Pymongo. Isso é bem fácil, veja a instalação utilizando o aplicativo conda conforme explicamos anteriormente:

```
~/anaconda3/bin$ ./conda install -c conda-forge Scrapy=1.3.2
```

instalando o Pymongo

```
~/anaconda3/bin$ ./conda install pymongo
```

Pronto! temos as bibliotecas Python instaladas em poucos minutos :)

Caso você não queira usar o Anaconda abaixo estão os passos para instalação de cada uma.

Se você estiver usando um ambiente Linux é bem provável que o Python já virá instalado na sua distribuição.

No nosso exemplo, usamos o [Ubuntu](#). Neste, o **Python** já vem instalado na versão 2.7.12. Veja:

```
python --version  
Python 2.7.12
```

Caso não tenha instalado, use o gerenciador de pacotes da sua distribuição, como no exemplo abaixo:

```
sudo apt-get install python
```

No momento em que escrevo esse ebook, o Python se encontra na versão 3.6.0

É nessa versão que vamos trabalhar aqui, mas fique a vontade para usar a versão que melhor atenda você. Para instalar o Python3 faça:

```
sudo apt-get install python3-all
```

Após isso é chamar o python3 no terminal como:

```
python3
```

Caso tenha alguma dúvida, consulte a documentação oficial [aqui](#).

Agora que temos o Python instalado, vamos começar instalando a ferramenta [Jupyter Notebooks](#). Esta é sem dúvida uma das **melhores ferramentas** para se trabalhar com Data Science. É uma verdadeira mão na roda, pois, através de um browser podemos emitir instruções python, plotar gráficos ou manipular dados de uma forma intuitiva e simples. Tudo isso de forma rápida e sem ter que escrever tantos programas.

Com os Notebooks seus scripts ficam documentados e é fácil de compartilhar com outros programadores. Para instalar o **Jupyter Notebooks** faça:

Instale a ferramenta pip, se esta já não estiver instalada:

```
sudo apt install python-pip
```

Em seguida instale o jupyter.

```
sudo pip install jupyter
```

Pronto, com o jupyter instalado, inicie a aplicação para ter certeza que não houve erro na instalação

```
jupyter notebook
```

Deve ser exibido o browser com a interface do jupyter notebooks.

Numpy

O [NumPy](#) como já dito anteriormente, é uma biblioteca muito usada, pois, facilita bastante operações matemáticas e lógicas como ordenação, seleção, transformações, operações estatísticas básicas. Para instalar faça:

```
sudo pip install numpy
```

O processo é bem simples e rápido. Em poucos segundos a biblioteca já está instalada e pronta para uso. Para testar se tudo ocorreu bem, faça um teste simples, veja:

```
python
```

Importe a biblioteca e execute o bloco de código como no exemplo abaixo:

```
import numpy as np
a = np.arange(6)
print (a)
[0 1 2 3 4 5]
```

Caso você tenha recebido a saída mostrada acima, a instalação foi bem sucedida.

Matplotlib

O [Matplotlib](#) é uma biblioteca python usada para plotagem de gráficos 2D. O objetivo desta é descomplicar a plotagem de gráficos e tornar fácil a visualização de dados. Para instalar faça:

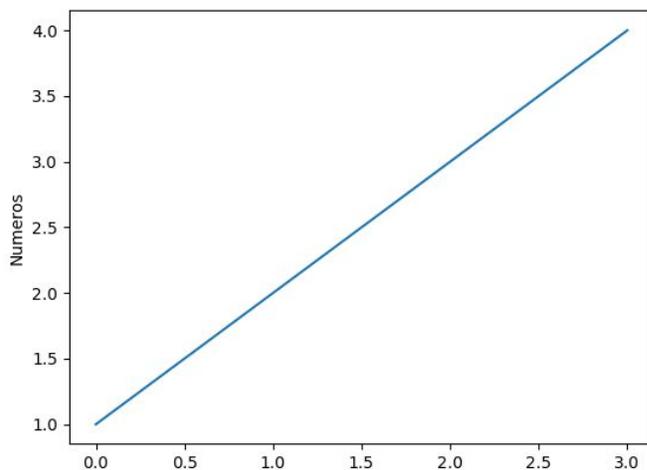
```
sudo pip install matplotlib
```

Aguarde um pouco até a conclusão da instalação. Para testar se tudo ocorreu bem, faça um teste simples, veja:

```
python
```

Importe a biblioteca e execute o bloco de código como no exemplo abaixo:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('Numeros')
plt.show()
```



O código acima deverá plotar um gráfico simples na tela conforme a imagem acima. Se deu tudo certo, sua instalação está Ok.

Pandas

A [pandas](#) é uma biblioteca open source bastante poderosa, de forma intuitiva e fácil fornece recursos para trabalhar com estruturas de dados com um ótimo desempenho. Particularmente é a biblioteca que eu mais gosto ;). Para instalar execute o comando abaixo:

```
sudo apt-get install python-pandas
```

Aguarde a instalação da biblioteca. Esta deve demorar mais um pouco, pois, possui diversas dependências. Para testar se tudo ocorreu bem, faça um teste simples, veja:

```
python
```

Importe a biblioteca e execute o bloco de código como no exemplo abaixo:

```
import pandas as pd  
num = pd.Series([1,3,5,6,"8"]  
print (num)  
0 1  
1 3  
2 5  
3 6  
4  
5 8  
dtype: object
```

Se a saída acima foi impressa na tela, sua instalação está ok.

Scikit-learn

A [Scikit-learn](#) é uma biblioteca simples e eficiente para trabalhar com Machine Learning. Esta contém diversos algoritmos, métodos e utilitários já implementados em Python que podem ser executados facilmente. Para instalar faça:

```
sudo pip install scikit-learn
```

Aguarde alguns instantes e pronto já está instalada e pronta para trabalhar. Para testar se tudo ocorreu bem, faça um teste simples, veja:

```
python
```

Importe a biblioteca e execute o bloco de código como no exemplo abaixo:

```
from sklearn import datasets
iris = datasets.load_iris()
print (iris)
```

Este deve imprimir a estrutura do dataset Íris. Se foi impresso sem qualquer erro, a biblioteca foi instalada com sucesso.

NLTK

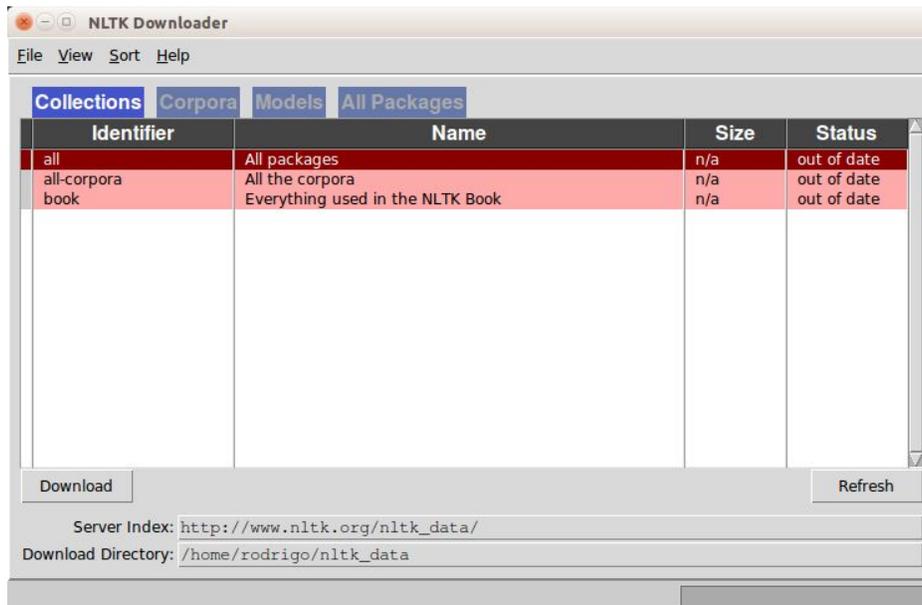
o [NLTK](#) é um kit de ferramentas para processamento da linguagem natural. Com este é possível trabalhar com técnicas de NLP de forma fácil e intuitiva, pela sua vasta quantidade de métodos existentes, o NLTK é um kit obrigatório quando o assunto é mineração de textos ou linguística. Para instalar faça:

```
sudo pip install nltk
```

Após instalar, conecte no console e faça o download do corpus da NLTK. Este contém diversos datasets para testes, textos e é bem útil. Eu normalmente faço download de tudo.

```
import nltk
nltk.download()
```

Será exibida a imagem abaixo. Clique em "all" e em OK e aguarde o download.



R - Simplicidade e Eficiência!



R é uma linguagem de programação extremamente poderosa e que tem um espaço em destaque quando o assunto é Data Science. Famosa pela sua facilidade para fazer análise de dados, processar instruções estatísticas e modelos gráficos.

Para instalar o R, primeiro a ser feito é descobrir codinome do Ubuntu, para isso utilize o seguinte comando:

```
lsb_release -a  
Codename: trusty
```

No site do R tem a lista de repositórios que mantém o projeto. Escolha o repositório de onde você quer fazer o download. No meu caso escolhi o [repositório](#) da USP no Brasil

Importante: Deve haver um espaço entre o link e o nome da versão do seu ubuntu

Adicione o endereço do repositório e o codinome do seu Ubuntu no final do arquivo `/etc/apt/source.list`. No meu caso ficou:

```
vi /etc/apt/source.list  
deb http://vps.fmvz.usp.br/CRAN/bin/linux/ubuntu trusty/
```

Em seguida adicione as chaves

```
sudo gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9  
sudo gpg -a --export E084DAB9 | sudo apt-key add -
```

Agora atualize a lista de repositórios da sua máquina e instale os pacotes do R

```
sudo apt-get update  
sudo apt-get install r-base r-base-dev
```

Após instalação abra o console do R com o comando “R” (maiúsculo):
R

Pronto! R Instalado

R-Studio

Quem trabalha com R sabe que existem diversas ferramentas para ajudar o desenvolvedor a trabalhar com essa linguagem. A ferramenta que mais se destaca é o **R-studio**.

Esta é uma IDE open source para R muito útil pois facilita bastante o desenvolvimento. Nesse ebook vamos instalar a versão **R-Studio Desktop** dessa ferramenta.

Para instalar o R-Studio no Ubuntu faça download do pacote [aqui](#) .No meu caso precisei instalar a seguinte biblioteca, pois, essa é uma dependência para instalação do R-Studio.

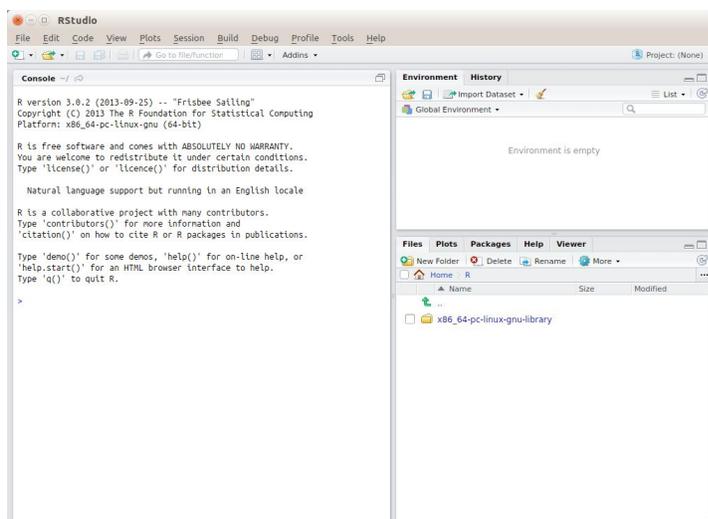
```
sudo apt-get install libjpeg62
```

Em seguida instale o R-Studio

```
sudo dpkg -i rstudio-1.0.136-amd64.deb
```

Pronto! Chame o **R-Studio** no console com o comando:

```
rstudio
```



Se estiver usando **Windows**, a instalação do R é bem fácil, basta fazer o download [aqui](#) e seguir o assistente de instalação.

RapidMiner - Acelere suas análises através de Workflows



RapidMiner como o nome já diz é uma plataforma para trabalhar com DataScience de forma rápida, simples e visual.

As ferramentas oferecidas fornecem uma interface gráfica rica com objetos e processos que simplificam as diversas tarefas do dia a dia de um cientista de dados.

Através do RapidMiner Studio é possível criar workflows extremamente intuitivos com objetos que executam todas as diversas tarefas, como, leitura e carregamento dos dados, limpeza e transformação, filtragem, modelagem, aplicação de algoritmos de **Machine Learning** e visualização dos resultados.

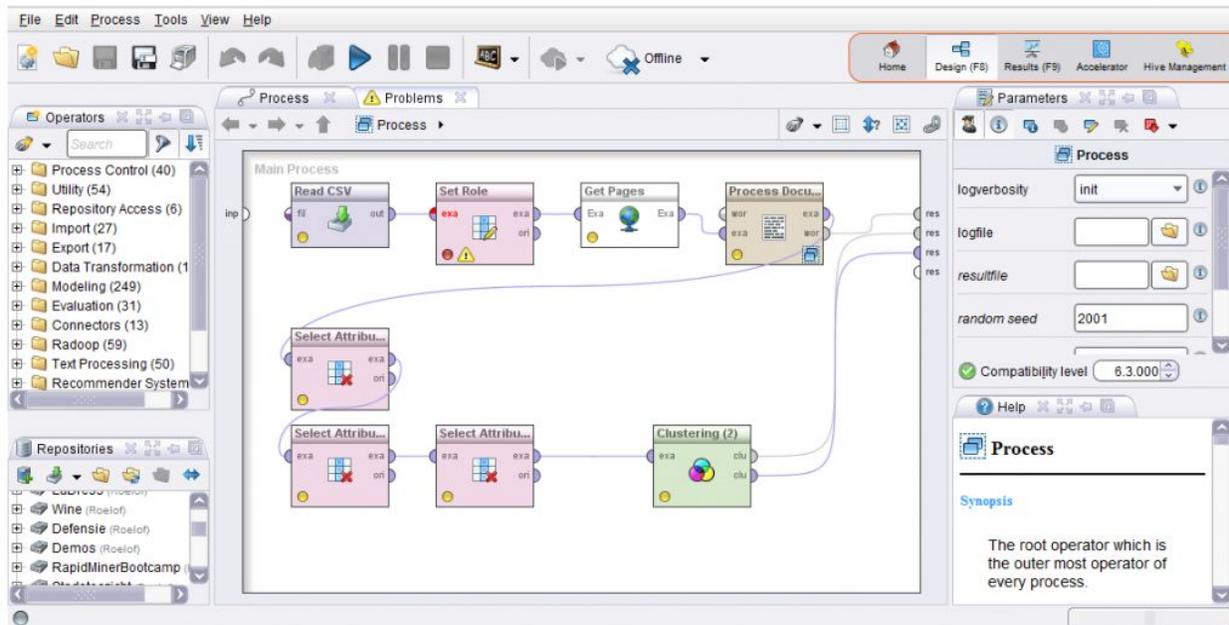
O diferencial do **RapidMiner** é a facilidade e velocidade para criar modelos preditivos já que não é necessário o trabalho de codificação e transformação dos dados. Dessa forma o processo de validação e ajuste do modelo se torna simples.

Os três produtos oferecidos são o **RapidMiner Studio**, **RapidMiner Server** e **RapidMiner Radoop**.

- **RapidMiner Studio:** Utilizado para desenhar os Workflows que mapeiam todo o processo de mineração de dados desde o carregamento dos dados até a visualização dos resultados.
- **RapidMiner Server:** Utilizado para gerenciar seus modelos, compartilhar com outros usuários.
- **RapidMiner Radoop:** Utilizado para compilar e executar workflows armazenados no Hadoop.

A plataforma oferece um tipo de licenciamento gratuito que permite a utilização do **Rapidminer Studio** com uma base de dados de até 10 mil registros.

Veja na imagem abaixo um exemplo de **Workflow** criado no **RapidMiner**.



Como você pode ver é bem simples usar o Rapidminer. Como o Rapidminer simplifica vários passos, o desenvolvedor não precisa instalar diversas ferramentas para realizar seu trabalho.

Já esta tudo pronto, em um só lugar.

Weka - O Poder da GUI!



Weka é um projeto open source que significa **Waikato Environment for Knowledge Analysis** - Ambiente para Análise de Conhecimento Waikato. Foi criado como um projeto de Machine Learning pela universidade de Waikato na Nova Zelândia.

O projeto tem o objetivo de disseminar técnicas de **Machine Learning** através da disponibilização do software para utilização de pesquisadores, alunos e para resolver problemas reais da indústria além de contribuir com a ciência pela mundo.

O grande diferencial do **Weka** além de todo o seu arsenal de métodos e algoritmos é a sua interface gráfica (GUI - Graphical User Interface) que torna as tarefas de mineração de dados extremamente fáceis e rápidas.

Através da interface é possível consultar dados em sistemas de bancos de dados, executar métodos de processamento de dados, executar e configurar parâmetros dos algoritmos e visualizar os resultados através de gráficos. Tudo isso sem precisar escrever comandos ou programar.

O Weka tem funcionalidades para manipulação de bases de dados (pré-processamento), interface para visualização de dados, e ainda disponível diversos algoritmos de machine learning e Data Mining. Isso facilita muito a vida dos seus usuários que não tem que dominar diversas ferramentas para fazer seu trabalho.

Para quem gosta de escrever comandos ou programar scripts o **Weka** fornece também acesso a sua vasta coleção de técnicas e algoritmos via API.

Dessa forma podemos utilizar seus recursos em programas **Java**. Consulte a documentação [aqui](#).

O Weka é uma ferramenta desenvolvida em **Java** e pode ser baixado e utilizado livremente em diferentes plataformas como Windows, Linux e Mac.

Instalando o Weka

Para instalar e executar o Weka é super simples. O primeiro passo é fazer o download neste [link](#) conforme seu sistema operacional. Observe que para o Weka funcionar é necessário a instalação do ambiente de execução do Java (JRE). No Ubuntu para instalar o JRE, basta instalar o pacote o **default-jre** conforme o comando abaixo:

```
sudo apt-get install default-jre
```

Após instalar o Java, descompacte o pacote do Weka, e em seguida, execute sua interface gráfica como no exemplo abaixo:

```
unzip weka-3-8-1  
cd weka-3-8-1/  
java -jar weka.jar
```

Será exibida a interface como na imagem abaixo:



Pronto, a instalação foi feita com sucesso!

Para o ambiente **Windows** é possível baixar o Weka juntamente com o Java, após isso é só seguir o assistente de instalação normalmente.

Conclusão

Neste capítulo vimos as ferramentas e bibliotecas **essenciais** para se trabalhar com Data Science.

Começamos com a linguagem **Python** e todas as suas bibliotecas que facilitam o trabalho além de oferecer um grande apoio da comunidade por ser uma tecnologia extremamente disseminada.

Vimos como instalar o pacote **Anaconda** e também as bibliotecas individualmente. Logo após vimos como instalar o **R** e sua IDE de desenvolvimento **RStudio** para trabalhar com análise de dados, além disso conhecemos e instalamos os projetos **Weka** e **RapidMiner** e vimos como essas ferramentas são poderosas para realizar tarefas de Data Science de forma rápida e produtiva.

Existem diversas ferramentas, bibliotecas e plataformas para trabalhar com **Data Science**. Saber utilizá-las de forma correta, pode trazer diversos benefícios para o Cientista de Dados como produtividade, clareza e facilidade.

Dominando o Python: A Linguagem Fundamental para Trabalhar com Data Science



Como falamos acima, Python é uma linguagem fenomenal. Dominar essa linguagem é de suma importância para qualquer pessoa que quer trabalhar com projetos de Data Science.

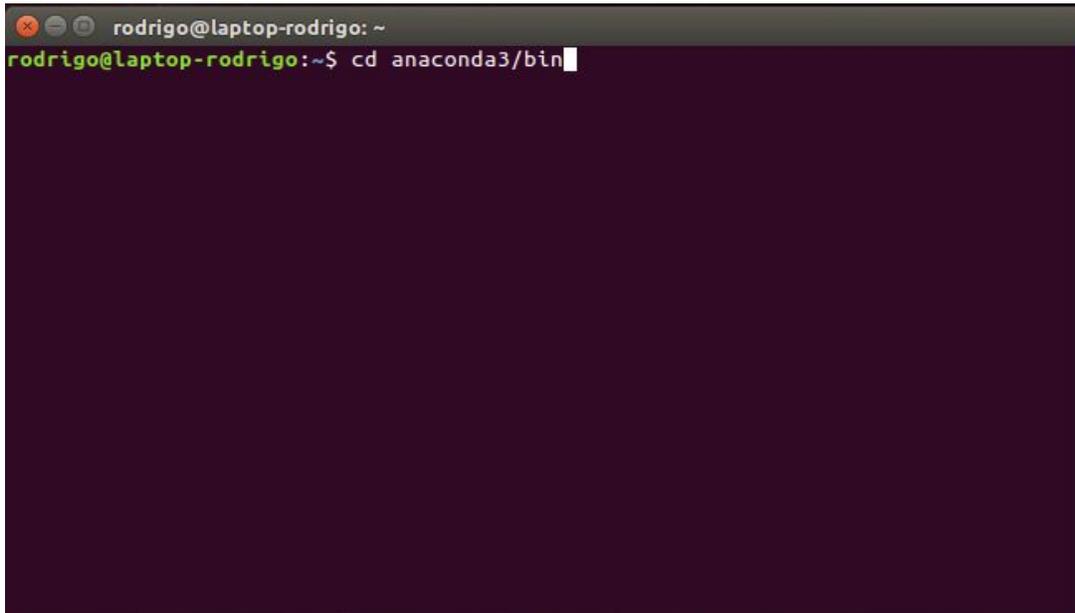
Neste capítulo vamos conhecer um pouco mais sobre essa linguagem, como manipular seus operadores, como a linguagem funciona e como fazer tarefas essenciais.

Para fazer os exemplos deste capítulo vou utilizar a ferramenta Jupyter Notebooks. Como já explicado anteriormente, se você já instalou a plataforma Anaconda, o Jupyter já vem pronto.

Você vai ver que essa ferramenta é extremamente fácil de usar e muito útil.

Inicie o terminal (se estiver usando linux) e execute a ferramenta.

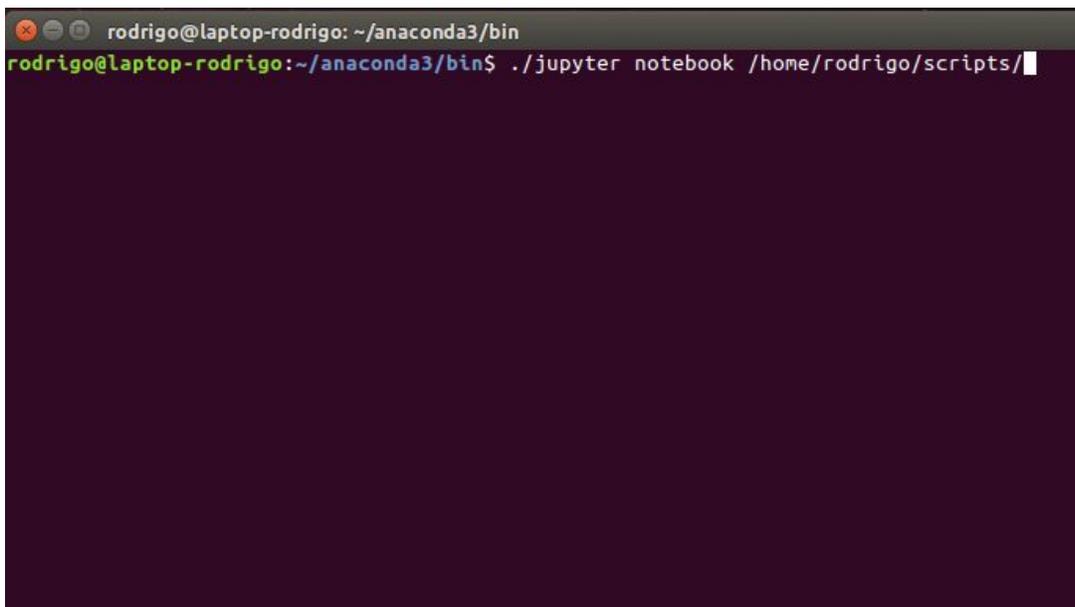
O primeiro a ser feito navegar até o diretório de instalação do anaconda e executar a jupyter:



```
rodrigo@laptop-rodri... ~  
rodrigo@laptop-rodri...~$ cd anaconda3/bin
```

Agora execute o utilitário **jupyter** passando o parâmetro “notebook” e o caminho do diretório que você quer utilizar para armazenar os seus scripts.

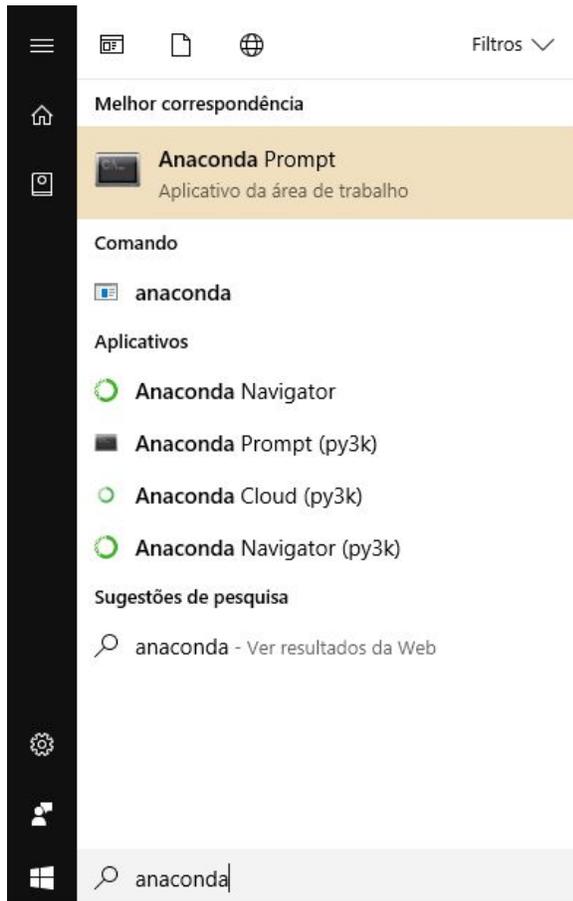
No meu caso criei o diretório “**scripts**” dentro da pasta home do usuário Rodrigo.



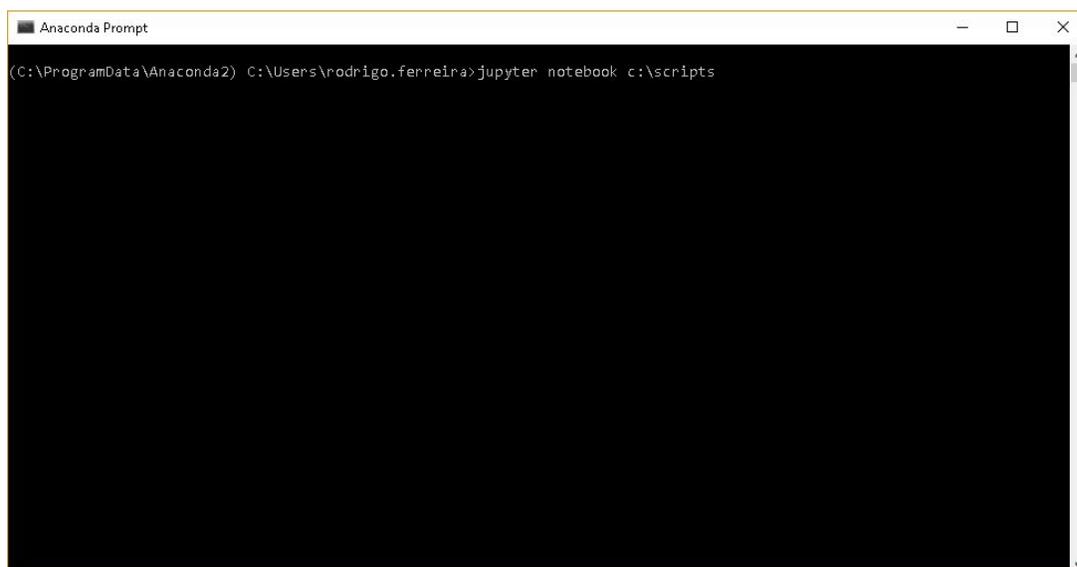
```
rodrigo@laptop-rodri... ~/anaconda3/bin  
rodrigo@laptop-rodri...~/anaconda3/bin$ ./jupyter notebook /home/rodrigo/scripts/
```

Se você estiver usando **Windows** é muito simples também, basta chamar o prompt do Anaconda, veja:

Clique em Iniciar e digite anaconda



Como prompt aberto, chame o utilitário jupyter passando o parâmetro “notebook” e o diretório onde irá armazenar os scripts, veja:

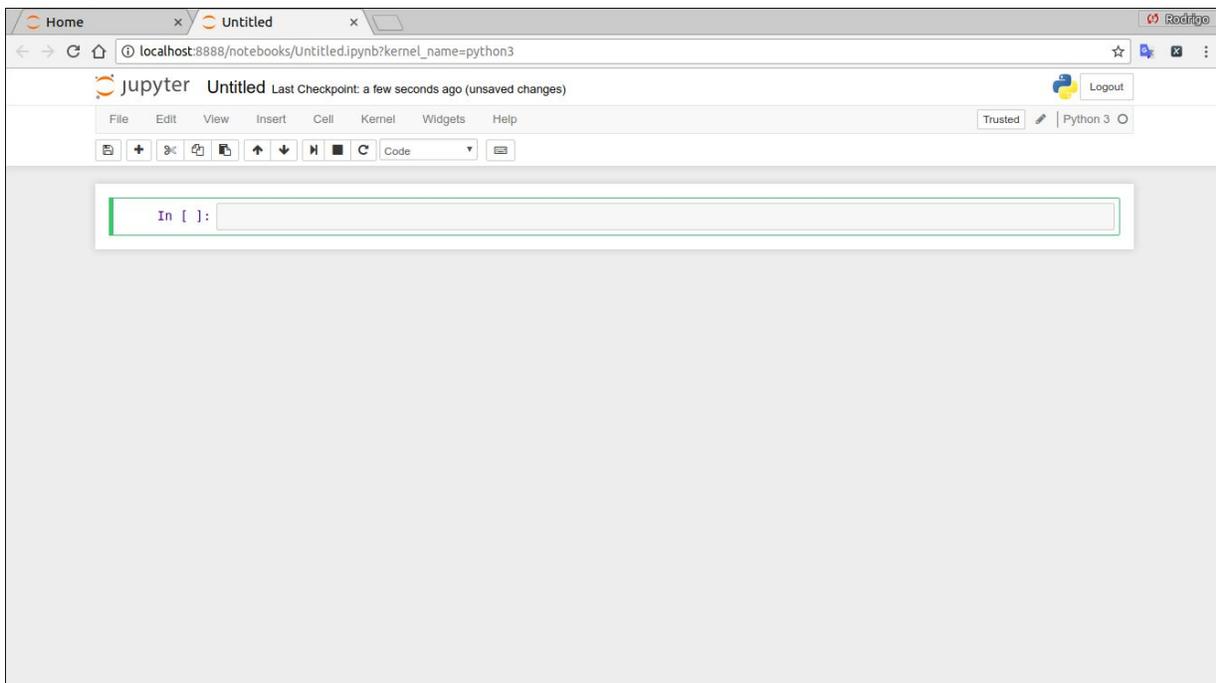


O jupyter abre o seu browser padrão e exibe a página que contém os seus notebooks. Como não temos nenhum notebook ainda, a página exibe vazia.

Vamos criar um notebook para conter nossos scripts. Clique em **“New”** como no exemplo abaixo:



Em seguida clique em **“Python 3”** e a tela abaixo deverá ser exibida



Pronto!!

O nosso notebook foi criado e já podemos começar escrever scripts.

Perceba que já temos uma célula selecionada, chamamos de células esse espaço que está com verde na borda.

Dentro da célula podemos inserir comandos python e executa-las independentemente. Veja um exemplo:

```
In [4]: print ("Olá Mundo!!")  
Olá!!
```

No exemplo acima rodei um comando python simples, basta colocar o comando e dar um **CTRL + ENTER** para executar.

Para inserir mais uma célula, clique em Insert e escolha **"Insert Cell Below"**. Insira mais algumas células para acompanhar os exemplos a seguir.

Operações Matemáticas

No python, fazer operações matemáticas é muito simples, veja alguns exemplos:

```
In [4]: print ("Olá Mundo!!")  
Olá!!
```

```
In [5]: 2 + 2  
Out[5]: 4
```

```
In [6]: 2 - 2  
Out[6]: 0
```

```
In [7]: 2 * 2  
Out[7]: 4
```

```
In [8]: 2 / 2  
Out[8]: 1
```

Veja que fiz as quatro operações básicas, soma, subtração, multiplicação e divisão. Tudo de forma simples e sem escrever muito.

Também é possível fazer outras duas operações facilmente, que são a potência e o módulo, faça você e veja o resultado:

```
2 ** 2
```

```
8 % 2
```

Números Float

Números float são números reais, ou seja, números fracionados. Veja alguns exemplos de operações com números float no python

```
In [9]: 3.4 + 6.6
```

```
Out[9]: 10.0
```

No exemplo acima fizemos a soma de dois números do tipo float. As operações mostradas acima também podem ser feitas da mesma forma com esse tipo de dado.

Funções

O Python contém diversas funções embutidas, vejamos algumas funções mais úteis:

Type: A função **type()** imprime o tipo de uma variável ou de um valor. Veja, no caso de alguns números, ao usar a função **type()** esta irá nos retornar que tipo está esse dado:

```
In [11]: type(5)
```

```
Out[11]: int
```

```
In [13]: type(6.5)
```

```
Out[13]: float
```

Funções de Conversão

È possível converter variáveis ou valores usando as funções de conversão. Veja um exemplo, de algumas conversões:

Converte um número float para inteiro

```
In [14]: int(6.0)
```

```
Out[14]: 6
```

Converte um número float para inteiro, porém, é realizado um arredondamento do valor

```
In [15]: int(6.5)
```

```
Out[15]: 6
```

Converte um número inteiro para float

```
In [16]: float(6)
```

```
Out[16]: 6.0
```

As funções são bem úteis, só devemos tomar cuidado no caso de arredondamento de valores, pois, em alguns casos pode haver uma perda da informação.

Por falar em arredondamento, temos uma função chamada `round()` que faz essa tarefa, veja:

```
In [19]: round(2.456578,2)
```

```
Out[19]: 2.46
```

Por padrão a função `round()` elimina qualquer valor após a vírgula. É possível passar como parâmetro para a função, a quantidade de números que devem ser mantidos após a vírgula.

No exemplo abaixo, a função irá arredondar o valor e manter apenas 2 números após a vírgula, veja:

```
In [19]: round(2.456578,2)
```

```
Out[19]: 2.46
```

E se quisermos manter 3 números? É só passar por parâmetro para a função:

```
In [18]: round(2.456578,3)
```

```
Out[18]: 2.457
```

Variáveis e Strings

As variáveis no python são de tipagem dinâmica, ou seja, o interpretador define o tipo de dados conforme a variável recebe dados, sem a necessidade de que o desenvolvedor tenha que definir o tipo anteriormente.

Por exemplo, veja a declaração e execução das variáveis no python, veja:

```
In [21]: var1 = 1  
var2 = 3  
var1 + var2
```

```
Out[21]: 4
```

No exemplo acima, criei duas variáveis “var1” e “var2” e somei seus valores, como feito anteriormente.

Como falei, no python a tipagem dinâmica me dar a liberdade de atribuir um outro tipo de dados a qualquer uma dessas variáveis, veja o exemplo abaixo:

```
In [24]: var1 = "Rock Forever"  
var1
```

```
Out[24]: 'Rock Forever'
```

Atribui uma frase a variável “var1”, ou seja, agora a variável var1 é do tipo **string**. Veja com o comando type:

```
type(var1)
```

Manipulação de strings



Trabalhar com strings no Python é muito simples, veja que a linguagem já tem recursos nativos que ajudam em muito o desenvolvedor. Como concatenar sentenças ? é fácil:

```
In [25]: frase = 'Python eh uma linguagem fenomenal'
         frase2 = ' para qualquer cientista de dados'
         frase + frase2
```

```
Out[25]: 'Python eh uma linguagem fenomenal para qualquer cientista de dados'
```

O python irá concatenar as duas string. E com listas como funciona ? mesma coisa:

No caso de listas você pode ainda usar o método **append** para adicionar apenas um elemento por exemplo:

```
In [27]: lista = ['Segunda', 'Terça', 'Quarta']
```

```
In [28]: lista.append('Quinta-feira')
```

```
In [29]: lista
```

```
Out[29]: ['Segunda', 'Terça', 'Quarta', 'Quinta-feira']
```

Ainda com strings podemos varrer uma frase imprimindo as posições dos caracteres, veja:

```
In [32]: frase2[0:15]
```

```
Out[32]: ' para qualquer '
```

No caso das listas, se quisermos imprimir do quarto elemento em diante ? faça o teste :

```
lista[3:8]
```

Obs: Lembrando que os índices das listas iniciam com 0.

Objeto String

É muito comum contar quantos termos tem em uma sentença ou um texto. Por exemplo, em nossa frase acima, quantas palavras contém na frase?

len(lista)

A função `len()` irá contar quantos elementos tem na lista, como temos as palavras na lista, é só usar a função `len` do python. Agora imagine que você precisa contar quantas ocorrências aconteceram para a palavra “six” na frase “*Six six six the number of the Beast*”

```
In [34]: Frase2 = 'Six six six the number of the Beast'  
Frase2 = Frase2.lower()  
Frase2.count('six')
```

```
Out[34]: 3
```

Na linha 2 a função `lower()` coloca todas as palavras da frase em minúsculo, assim contamos quantas ocorrências existem para a palavra ‘six’ na linha 3 com o método `count()`.

E se você quiser contar quantos caracteres temos na frase acima:

```
In [35]: len(frase2)
```

```
Out[35]: 33
```

Em caso de uma string a função `len()` conta quantos caracteres tem essa string, ou seja, seu tamanho. Outro recurso interessante quando estamos trabalhando com Strings é a função `index`, essa permite encontrarmos o índice de uma palavra ou caractere em uma frase, veja um exemplo:

Imagine que quero saber o índice onde se começa a palavra ‘cientista’:

```
In [36]: frase2.index('cientista')
```

```
Out[36]: 15
```

Com o valor retornado, sem a posição na frase, ou seja, o índice. Agora quero varrer os outros caracteres e imprimir do 14 até o 35, veja:

```
In [38]: frase2[14:35]
```

```
Out[38]: 'cientista de dados'
```

A função `join` é bem útil também. Com esta podemos fazer uniões, veja um exemplo onde quero unir dois elementos de uma lista e transformá-los em uma única string:

```
In [39]: lista=['hot','Dog']  
         ' '.join(lista)
```

```
Out[39]: 'hot Dog'
```

Obs: Um espaço entre as aspas antes do join vai unir os elementos com o espaço em branco entre eles

Listas

No Python temos algumas estruturas de dados. Vamos falar aqui das Listas e dos Dicionários

A listas são tipos de dados bem versáteis, podem conter dados de tipos diferentes, podem ser aninhadas e são bem fáceis de manipular. Veja um exemplo de uma lista abaixo:

```
In [51]: list = ['dog', 'blues', 123456, 10.9]  
         list
```

```
Out[51]: ['dog', 'blues', 123456, 10.9]
```

Perceba que criamos uma lista chamada 'list' e que esta contém itens com tipos String, Inteiro e Float.

Para acessar valores na lista usamos um índice, ou seja, passamos qual posição da lista queremos acessar, veja:

```
In [42]: list[0]
```

```
Out[42]: 'dog'
```

```
In [43]: list[3]
```

```
Out[43]: 10.9
```

No primeiro exemplo, imprimimos o valor do primeiro elemento da lista e no segundo exemplo imprimimos o último valor da lista. Os índices sempre iniciam com o valor 0, onde o 0 significa o primeiro ítem da lista.

É possível fazer diversas operações nas listas, como atualização de itens, exclusão, concatenação, multiplicação de itens etc.

Veja algumas operações:

Adicionando novos itens a lista:

```
In [52]: list.append('horse')
list
Out[52]: ['dog', 'blues', 123456, 10.9, 'horse']
```

Atualizando o valor do primeiro item da lista:

```
In [53]: list[0]='cat'
list[0]
Out[53]: 'cat'

In [54]: list
Out[54]: ['cat', 'blues', 123456, 10.9, 'horse']
```

Removendo itens da lista:

```
In [55]: del list[1]
list
Out[55]: ['cat', 123456, 10.9, 'horse']
```

Contando a quantidade de itens da lista:

```
In [56]: len(list)
Out[56]: 4
```

Concatenando listas:

Vamos criar uma segunda lista chamada list2 e concatenar as duas:

```
In [59]: list2 = [1,2,3,4]
         list + list2
```

```
Out[59]: ['cat', 123456, 10.9, 'horse', 1, 2, 3, 4]
```

Multiplicando itens de uma lista:

```
In [60]: list * 2
```

```
Out[60]: ['cat', 123456, 10.9, 'horse', 'cat', 123456, 10.9, 'horse']
```

Retorna “True” caso um valor pertence a lista e “False” caso contrário:

```
In [62]: 3 in list2
```

```
Out[62]: True
```

```
In [63]: 3 in list
```

```
Out[63]: False
```

O Python contém algumas funções embutidas que são bem úteis quando estamos trabalhando com listas, veja:

Retorna o valor máximo e mínimo dos itens da lista:

```
In [64]: max(list2)
```

```
Out[64]: 4
```

```
In [67]: min(list2)
```

```
Out[67]: 1
```

Retorna a posição do item ‘horse’ na lista:

```
In [68]: list.index('horse')
```

```
Out[68]: 3
```

Remove item da lista:

```
In [69]: list2.remove(4)
list2
```

```
Out[69]: [1, 2, 3]
```

Dicionários

Os dicionários são estruturas de dados bastante útil. Basicamente num dicionário temos uma chave e um valor para cada chave, onde essas chaves são únicas, mas os valores podem se repetir.

Os valores das chaves podem ser de qualquer tipo, tais como inteiros, strings, listas etc.

Vamos ver alguns exemplos de dicionários e algumas operações envolvendo essa estrutura de dados.

Para criar um dicionário faça:

```
In [87]: dic = {'nome': 'ze', 'Sexo': 'M', 'idade': 50}
dic
```

```
Out[87]: {'Sexo': 'M', 'idade': 50, 'nome': 'ze'}
```

No exemplo acima, criamos um dicionário com 3 chaves, repare que a sintaxe é diferente das listas, este usa as chaves {} entre os itens e seus valores.

O nome do nosso dicionário é dic, e se quiséssemos acessar o item nome seria:

```
In [72]: dic['nome']
```

```
Out[72]: 'ze'
```

Importante: Para definir o dicionário usamos as chaves {}, mas para acessar as chaves do dicionário usamos os colchetes []

Atualizando um dicionário

Caso queira atualizar o valor de alguma chave do dicionário, é simples.

No exemplo abaixo, atualizei o valor da chave 'idade' para 65, veja:

```
In [88]: dic['idade']=65
```

```
In [75]: dic
```

```
Out[75]: {'Sexo': 'M', 'idade': 65, 'nome': 'ze'}
```

É interessante também adicionar mais chaves ao dicionário. Vamos adicionar uma nova chave ao dicionário, veja como ficará:

```
In [89]: dic['Endereco']='Bairro Capim Dourado, 44, 504'  
dic
```

```
Out[89]: {'Endereco': 'Bairro Capim Dourado, 44, 504',  
          'Sexo': 'M',  
          'idade': 65,  
          'nome': 'ze'}
```

Removendo elementos de um dicionário

É possível apagar elementos individuais de um dicionário ou ainda apagar todo o seu conteúdo.

Para remover uma chave, por exemplo a chave 'Sexo', use a instrução 'del'

```
In [90]: del dic['Sexo']  
dic
```

```
Out[90]: {'Endereco': 'Bairro Capim Dourado, 44, 504', 'idade': 65, 'nome': 'ze'}
```

Caso queira remover todas as chaves do dicionário, use o método "clear" do dicionário, veja:

```
In [93]: dic.clear()  
dic
```

```
Out[93]: {}
```

Removendo um dicionário

Pode ser necessário remover um dicionário. Fazendo isso, o objeto será removido da memória da máquina. Veja:

```
In [95]: del dic;
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-95-0acdfafd98de> in <module>()  
----> 1 del dic;  
  
NameError: name 'dic' is not defined
```

Como visto acima, o objeto “dic” não está mais definido, ou seja, este não existe mais em memória.

Operações com Dicionários

Existem alguns métodos úteis para trabalhar com dicionários. Estes métodos facilitam bastante o trabalho do dia a dia e evita retrabalho do desenvolvedor.

Veja alguns

keys()

Esse método como o próprio nome já diz, retorna as chaves do dicionário, veja:

```
In [97]: dic = {'nome': 'ze', 'Sexo': 'M', 'idade': 50}
         dic.keys()
```

```
Out[97]: ['idade', 'Sexo', 'nome']
```

values()

Esse método também bastante intuitivo, retorna os valores das chaves do dicionário:

```
In [98]: dic.values()
```

```
Out[98]: [50, 'M', 'ze']
```

items()

Esse método retorna a lista de elementos do dicionário em formato de chave: valor. Veja:

```
In [99]: dic.items()
```

```
Out[99]: [('idade', 50), ('Sexo', 'M'), ('nome', 'ze')]
```

copy()

Esse método faz uma cópia simples de um dicionário. Veja:

```
In [100]: dic2 = dic.copy()
         dic2
```

```
Out[100]: {'Sexo': 'M', 'idade': 50, 'nome': 'ze'}
```

O método retorna uma cópia do dicionário “dic” para dentro do objeto “dic2”. É interessante notar que o dicionário “dic2” não tem nenhuma relação com o dicionário “dic”.

Vamos alterar o valor de uma chave e ver como este se comporta:

```
In [101]: dic2['idade']=70  
dic2
```

```
Out[101]: {'Sexo': 'M', 'idade': 70, 'nome': 'ze'}
```

```
In [103]: dic
```

```
Out[103]: {'Sexo': 'M', 'idade': 50, 'nome': 'ze'}
```

Perceba que uma alteração no dicionário “dic2” foi totalmente independente do dicionário “dic”.

get()

O método get é interessante quando o desenvolvedor quer obter o valor de uma chave e caso não exista, retorna um valor definido, veja:

```
In [104]: dic.get('nome','NaoExiste')
```

```
Out[104]: 'ze'
```

No exemplo acima, obtemos o valor da chave nome. Mas caso essa chave não exista o método iria retornar ‘NaoExiste’, veja um exemplo:

```
In [105]: dic.get('cpf','NaoExiste')
```

```
Out[105]: 'NaoExiste'
```

Ao consultar uma chave que não existe o método retorna a mensagem que definimos acima.

Este é bem simples e pode ser bem útil.

setdefault()

Esse método é muito interessante. O que ele faz é retornar o valor de uma chave informada e caso não exista este irá inserir a chave e um valor padrão. Veja:

```
In [108]: dic.setdefault('Telefone', 'NaoInformado')
```

```
Out[108]: 'NaoInformado'
```

No caso acima a chave 'telefone' não existe, então será inserido no dicionário a chave 'telefone' e o valor definido acima, veja como ficou o dicionário:

```
In [109]: dic
```

```
Out[109]: {'Sexo': 'M', 'Telefone': 'NaoInformado', 'idade': 50, 'nome': 'ze'}
```

update()

Esse método atualiza um dicionário com as chaves e valores de outro dicionário. Caso as chaves não existam no dicionário a ser atualizado, estas serão inseridas no dicionário. Veja um exemplo:

```
In [111]: dic2 = {'nome': 'Rodrigo', 'Cidade': 'Belo Horizonte'}
          dic.update(dic2)
          dic
```

```
Out[111]: {'Cidade': 'Belo Horizonte',
           'Sexo': 'M',
           'Telefone': 'NaoInformado',
           'idade': 50,
           'nome': 'Rodrigo'}
```

No exemplo acima, definimos que o dicionário **dic2** teria as chaves **nome** e **cidade**. 'Nome' já existe no dicionário 'dic', então seu valor foi atualizado. A chave 'Cidade' como não existia no dicionário dic, então esta foi adicionada.

Conclusão

Que bom que você chegou até aqui, ficamos muito felizes em contribuir para o seu crescimento e agregar valor para a sua carreira.

Neste ebook além de esclarecer alguns conceitos e definições procuramos ressaltar um pouco das oportunidades e desafios do mundo **Data Science**, descrevendo algumas aplicações e

cenários onde é utilizada a tecnologia para resolver problemas complexos nas mais variadas áreas como **Saúde**, **Educação**, **Entretenimento** e **Tecnologia** etc. Exploramos em detalhes cada uma das áreas e discutimos ideias e como a aplicação de Data Science tem revolucionado estes setores por meio de empresas e startups inovadoras.

Aprendemos também sobre algumas ferramentas e linguagens de programação mais utilizadas pela comunidade para trabalhar com **Data Science** de forma eficiente levando em consideração suas vantagens e desvantagens e melhores práticas de acordo com cada cenário.

Descrevemos as ferramentas que oferecem maior retorno se tratando de produtividade devido a uma rica interface gráfica, baseada em workflows como o **RapidMiner** e **Weka**, vimos também as vantagens se tratando de flexibilidade, escalabilidade e performance quando optamos em utilizar as linguagens **R** e **Python**. Listamos e introduzimos de forma detalhada cada uma das bibliotecas Python mais poderosas para trabalhar com Data Science, aprendemos a instalação individual de cada uma, bem como, a utilização através da plataforma **Anaconda**.

Por fim aprendemos a instalação e configuração da ferramenta **Jupyter** para o desenvolvimento de scripts e documentações além de fazer uma completa explicação da linguagem Python e suas técnicas como funções, variáveis, métodos, operações matemáticas e manipulação de objetos.

Mais uma vez, **OBRIGADO** por nos acompanhar até aqui!

