

Trabalho 2 – Disciplina Mineração de Dados

Utilizando Python e R para Mineração de Dados

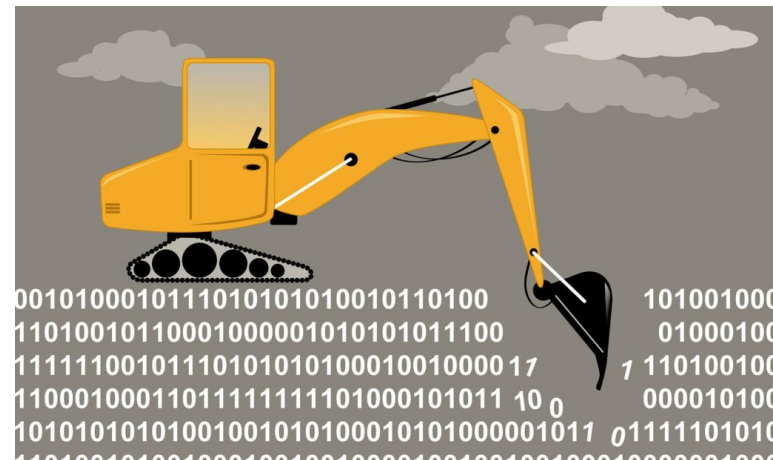


Universidade Federal Fluminense
Instituto de Computação
Aluno: Clayton Escouper das Chagas

Utilizando Python e R para Mineração de Dados

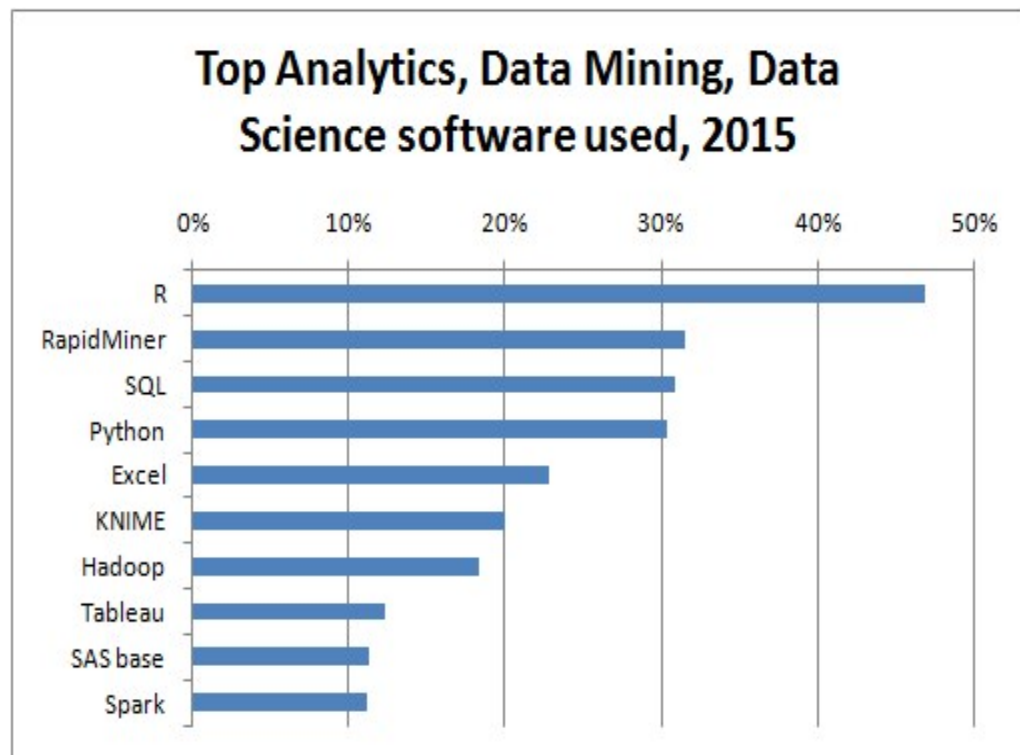
Sumário

1. Motivação
2. Python para Mineração de Dados
3. R para Mineração de Dados
4. Comparação
5. Demonstrações
6. Conclusão
7. Referências



1. Motivação

- Ferramentas de Mineração de Dados vs. Programação de Mineração de Dados



1. R*, 46.9% share (38.5% in 2014) /FO
2. RapidMiner**, 31.5% (44.2% in 2014) /C - Java
3. SQL*, 30.9% (25.3% in 2014)
4. Python*, 30.3% (19.5% in 2014) /FO
5. Excel**, 22.9% (25.8% in 2014) /C
6. KNIME**, 20.0% (15.0% in 2014) /FO - Java
7. Hadoop**, 18.4% (12.7% in 2014) /FO - Java
8. Tableau**, 12.4% (9.1% in 2014) /C - C++
9. SAS**, 11.3 (10.9% in 2014) /C - C
10. Spark**, 11.3% (2.6% in 2014) /FO - Java, Python, Scala e R
11. Weka*, 11.2% (17.0% in 2014) / FO - Java

Ferramentas de DM em ascensão (dados de 2016/2017): H2O/FO, Orange/FO, Mahout/FO

* - Linguagens

** - Ferramentas ou suites com DM

/FO – free e/ou open source

/C - comercial

Fonte: www.kdnuggets.com (16th annual KDNuggets Software Poll – mais de 3000 votos entre comunidade e empresas)

1. Motivação

- Ferramentas de Mineração de Dados

- > Vantagens:
 - >> Curva de aprendizado mais rápida
 - >> Mais completas (tarefas e completude)
 - >> Contratação de suporte
- > Desvantagens:
 - >> Custo (das mais completas)

- Programação de Mineração de Dados

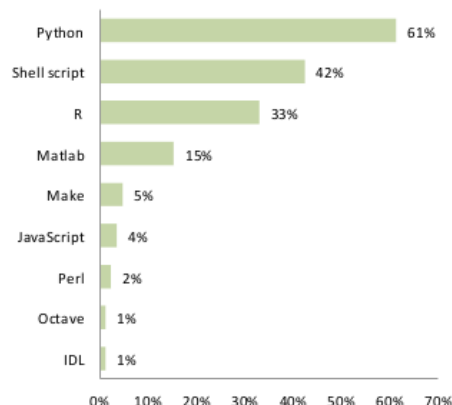
- > Vantagens:
 - >> Flexibilidade
 - >> Possibilidade de embarcar DM a softwares
 - >> Baixo custo
 - >> Simplicidade
- > Desvantagens
 - >> Curva de aprendizado mais lenta (...mas nem tanto :-)
 - >> Maior dificuldade de contratação de suporte

1. Motivação

- Utilização de linguagens de programação para experimentos científicos

61%

of the respondents*
have **Python** among
their preferred/more
often used tools to run
experiments




*Survey sent to AMC@UvA (Olabarriaga), UFRJ (Mattoso), DATAONE (newsletter), DBBras (mailing list), FIOCRUZ (Davila), USP (Traina), INRIA-Montpellier (Zenith group), LNCC (Ocana), PW 2016 TPC, SciPyLA (Telegram), Software Carpentry (mailing list), U. Nantes (Gaignard), UPENN (Davidson), receiving 85 answers.

- TIOBE Index

Nov 2017	Nov 2016	Programming Language	Ratings
1	1	Java	13.231%
2	2	C	9.293%
3	3	C++	5.343%
4	5	Python	4.482%
5	4	C#	3.012%
6	8	JavaScript	2.972%
7	6	Visual Basic .NET	2.909%
8	7	PHP	1.897%
9	16	Delphi/Object Pascal	1.744%
10	9	Assembly language	1.722%
11	19	R	1.605%
12	15	MATLAB	1.604%
13	14	Ruby	1.593%
14	13	Go	1.570%
15	10	Perl	1.562%
16	26	Scratch	1.550%

2. Python para Mineração de Dados

- Consigo implementar tudo de Mineração de Dados (pré-processamento, manipulação dos dados, algoritmos de mineração, visualização, etc) em Python “do zero”?
- R: Até dá...mas vai dar um trabalho!!! 
- Solução mais viável:
 - > **Caixa de Ferramentas Python para Mineração de Dados**

2. Python para Mineração de Dados

- Caixa de Ferramentas Python para Mineração de Dados (instalação + import das bibliotecas):

- > **Python 2.x** ou **3.x**: LP fácil, eficiente e rápida
- > **NumPy**: biblioteca de extensão para computação científica com suporte a arrays e matrizes multidimensionais, com funções matemáticas especializadas para estas estruturas
- > **SciPy**: biblioteca que estende as funcionalidades do NumPy (pré-requisito), para trabalhar de forma otimizada e com mais funções para manipulação e visualização de computação científica, principalmente manipulação de arrays multidimensionais
- > **Pandas**: pacote com estruturas especializadas para dados relacionais ou não estruturados, tornando esta manipulação mais fácil e flexível

2. Python para Mineração de Dados

- Caixa de Ferramentas Python para Mineração de Dados (instalação + import das bibliotecas):

> **Matplotlib**: biblioteca rica em recursos para construção de gráficos (plot), com API que dá suporte a GUI, Qt, GTK e OpenGL

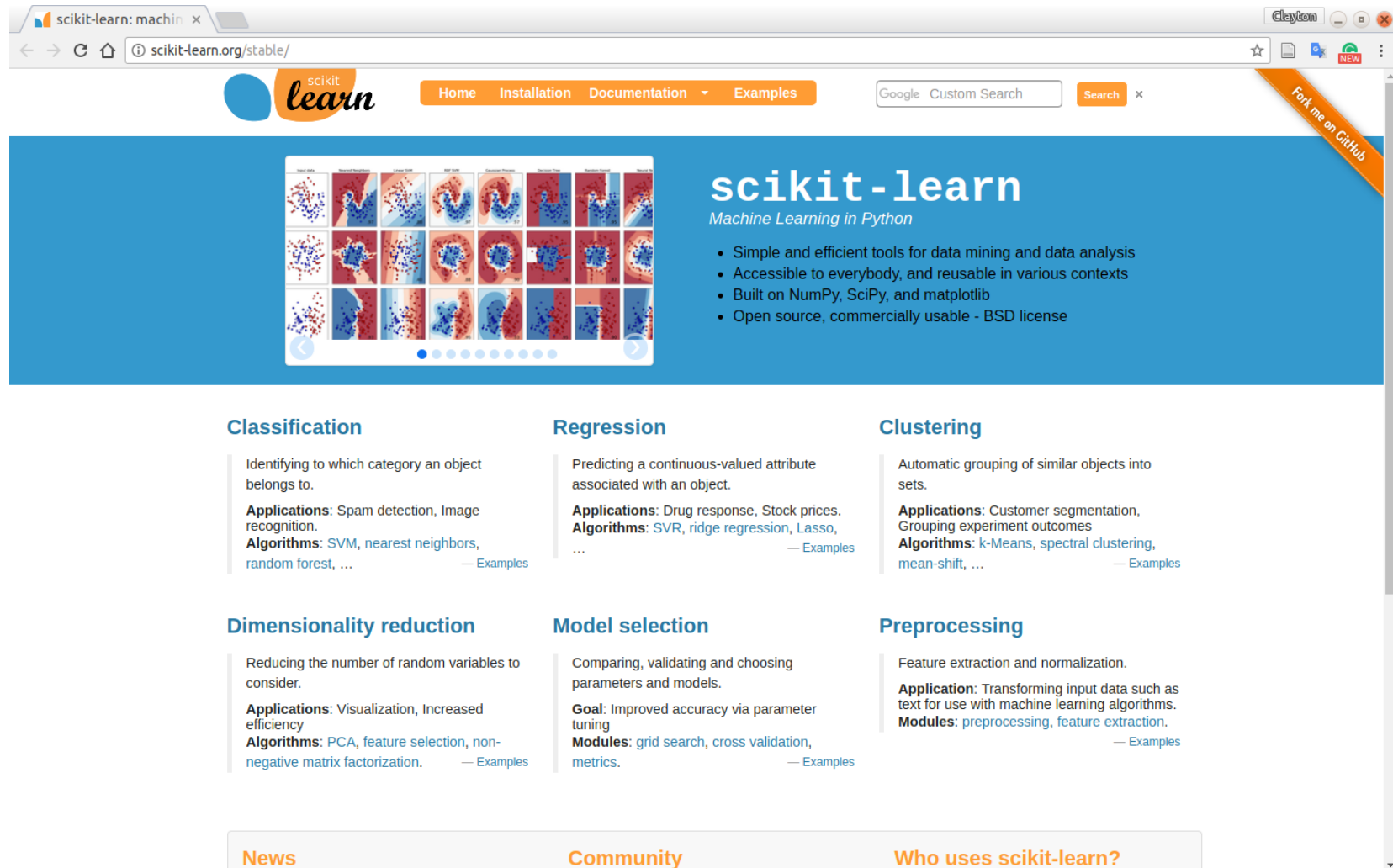
> **iPython**: pacote para programação/interpretação interativa via linha de comando

> **Jupyter**: extensão do iPython, com notebook interativo web

> **Scikit-learn**: (SciPy Toolkit) biblioteca padrão Python com implementações para Mineração de Dados e Aprendizado de Máquina

2. Python para Mineração de Dados

- Scikit-learn: v 0.19.0 (Ago17, desde Jun07)



The screenshot shows the scikit-learn website homepage. The browser address bar displays 'scikit-learn: machin x' and 'scikit-learn.org/stable/'. The website features a navigation bar with links for Home, Installation, Documentation, and Examples, along with a search bar. The main header includes the scikit-learn logo and the text 'Machine Learning in Python'. A large grid of 12 small plots illustrates various machine learning concepts. Below this, the website is organized into six columns, each representing a different machine learning task: Classification, Regression, Clustering, Dimensionality reduction, Model selection, and Preprocessing. Each column provides a brief description, applications, and algorithms. At the bottom, there are links for News, Community, and Who uses scikit-learn?.

scikit-learn
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification
Identifying to which category an object belongs to.
Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression
Predicting a continuous-valued attribute associated with an object.
Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering
Automatic grouping of similar objects into sets.
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction
Reducing the number of random variables to consider.
Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection
Comparing, validating and choosing parameters and models.
Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing
Feature extraction and normalization.
Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

[News](#) [Community](#) [Who uses scikit-learn?](#)

2. Python para Mineração de Dados

- Scikit-learn

- > Escrito em Python e Cython
- > Ampla documentação e suporte comunitário
- > Funções para:
 - >> Manipulação dos dados
 - >> Pré-processamento
 - >> Tratamento estatístico
 - >> Métricas (ROC, acurácia, matriz de confusão, etc)
 - >> Classificação (árvore de decisão, k-NN, random forest, Naive Bayes, etc)
 - >> Regressão
 - >> Clusterização (k-means, DBSCAN, etc)
 - >> Redes neurais

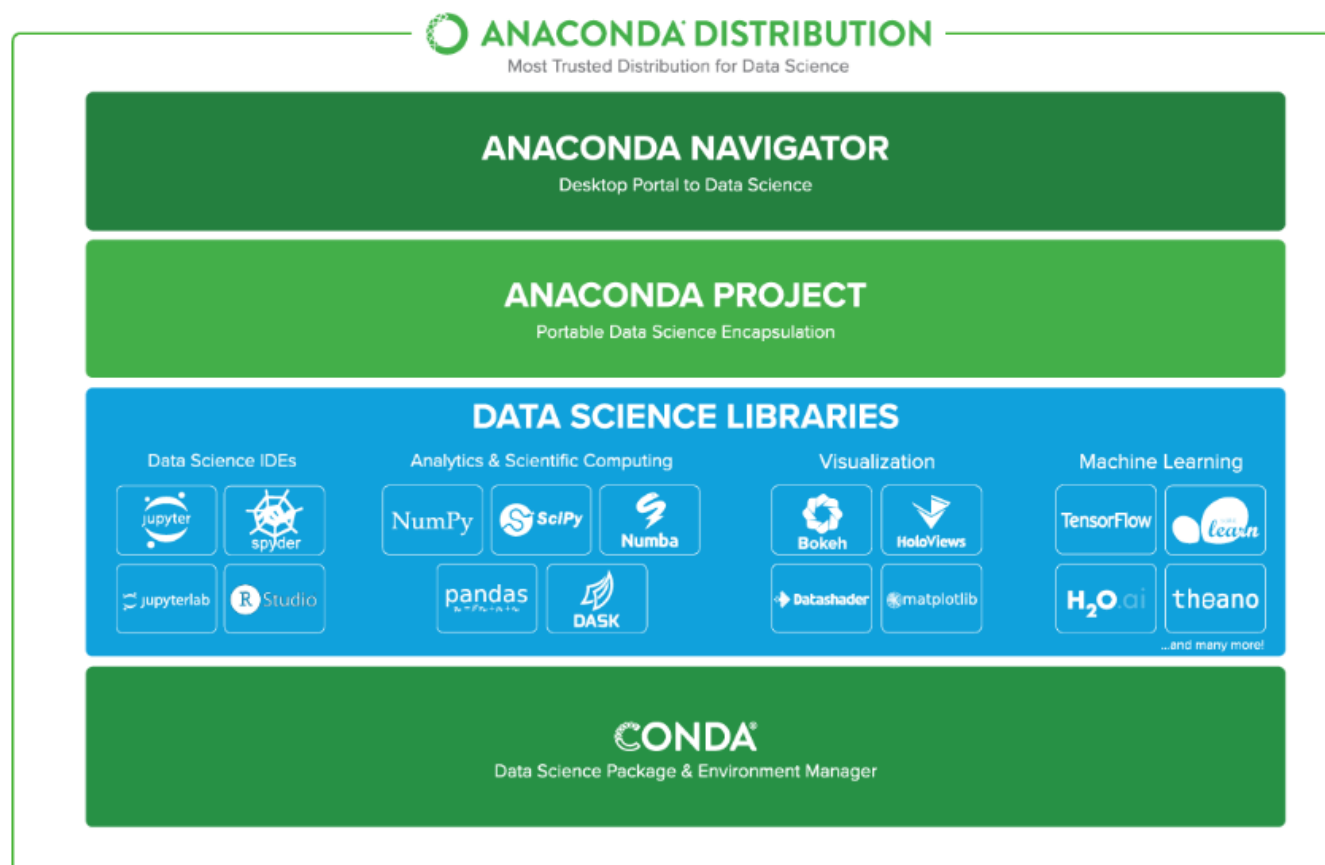
2. Python para Mineração de Dados

- Scikit-learn

5	API Reference	1183
5.1	sklearn.base: Base classes and utility functions	1183
5.2	sklearn.calibration: Probability Calibration	1189
5.3	sklearn.cluster: Clustering	1193
5.4	sklearn.cluster.bicluster: Biclustering	1231
5.5	sklearn.covariance: Covariance Estimators	1237
5.6	sklearn.cross_decomposition: Cross decomposition	1267
5.7	sklearn.datasets: Datasets	1281
5.8	sklearn.decomposition: Matrix Decomposition	1330
5.9	sklearn.discriminant_analysis: Discriminant Analysis	1383
5.10	sklearn.dummy: Dummy estimators	1391
5.11	sklearn.ensemble: Ensemble Methods	1396
5.12	sklearn.exceptions: Exceptions and warnings	1426
5.13	sklearn.feature_extraction: Feature Extraction	1431
5.14	sklearn.feature_selection: Feature Selection	1457
5.15	sklearn.gaussian_process: Gaussian Processes	1489
5.16	sklearn.isotonic: Isotonic regression	1527
5.17	sklearn.kernel_approximation: Kernel Approximation	1531
5.18	sklearn.kernel_ridge: Kernel Ridge Regression	1540
5.19	sklearn.linear_model: Generalized Linear Models	1543
5.20	sklearn.manifold: Manifold Learning	1642
5.21	sklearn.metrics: Metrics	1661
5.22	sklearn.mixture: Gaussian Mixture Models	1728
5.23	sklearn.model_selection: Model Selection	1739
5.24	sklearn.multiclass: Multiclass and multilabel classification	1795
5.25	sklearn.multioutput: Multioutput regression and classification	1803
5.26	sklearn.naive_bayes: Naive Bayes	1810
5.27	sklearn.neighbors: Nearest Neighbors	1821
5.28	sklearn.neural_network: Neural network models	1870
5.29	sklearn.pipeline: Pipeline	1884
5.30	sklearn.preprocessing: Preprocessing and Normalization	1892
5.31	sklearn.random_projection: Random projection	1936
5.32	sklearn.semi_supervised: Semi-Supervised Learning	1942
5.33	sklearn.svm: Support Vector Machines	1948
5.34	sklearn.tree: Decision Trees	1981
5.35	sklearn.utils: Utilities	2006
5.36	Recently deprecated	2019

2. Python para Mineração de Dados

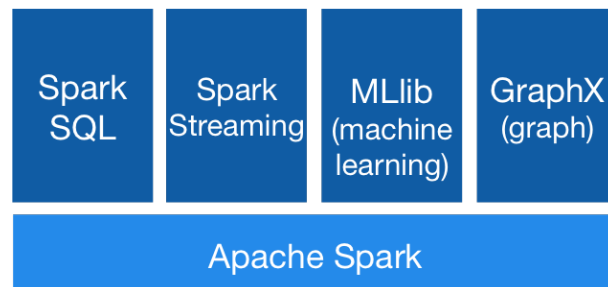
- **Anaconda:** distribuição com solução completa para Python e R *data science* (core, packages, IDEs, notebooks, gerenciador de pacotes, etc)



2. Python para Mineração de Dados

- MLlib

- > Módulo do Apache Spark com implementações de Mineração de Dados e Aprendizado de Máquina
- > Faz parte da arquitetura Spark (processamento de dados em grande escala)



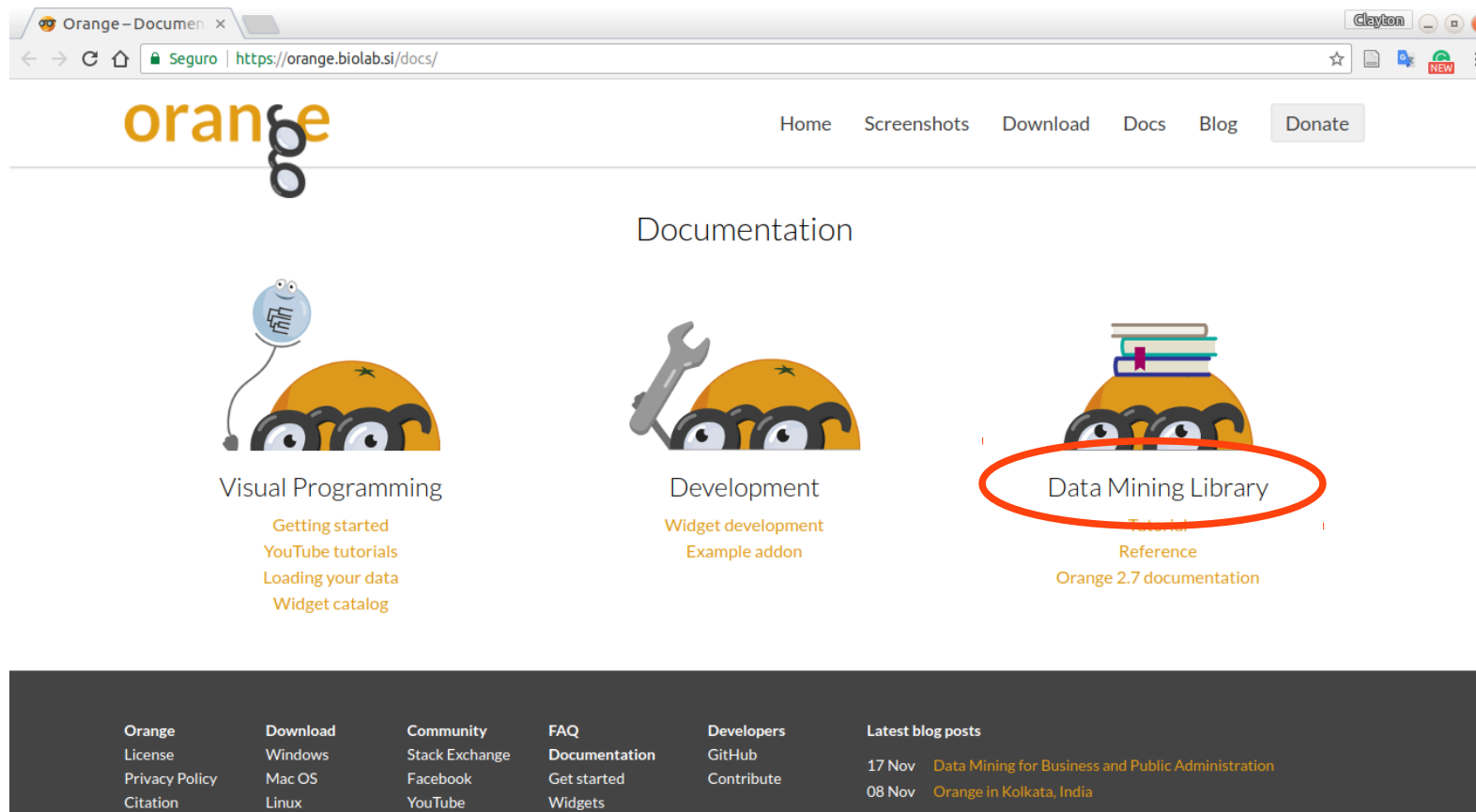
```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.DecisionTreeClassificationModel
import org.apache.spark.ml.classification.DecisionTreeClassifier
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}

// Load the data stored in LIBSVM format as a DataFrame.
val data = spark.read.format("libsvm").load("data/mllib/sample_libsvm_data.txt")
```

- Basic statistics
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

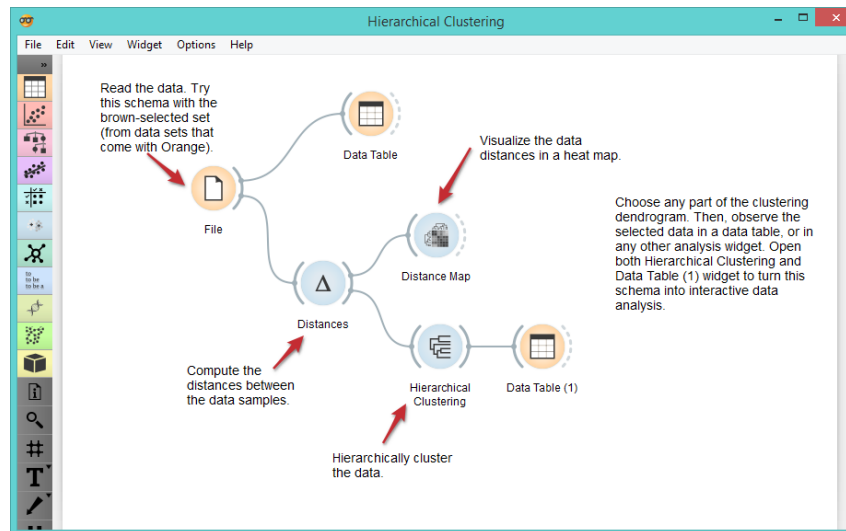
2. Python para Mineração de Dados

- Orange



2. Python para Mineração de Dados

- Orange



```
% python
>>> import Orange
>>> Orange.version.version
'3.2.dev0+8907f35'
>>>
```

If this leaves no error and warning, Orange and Python are properly installed and you are ready to continue with the Tutorial.

- **The Data**
 - ◊ [Data Input](#)
 - ◊ [Saving the Data](#)
 - ◊ [Exploration of the Data Domain](#)
 - ◊ [Data Instances](#)
 - ◊ [Orange Data Sets and NumPy](#)
 - ◊ [Meta Attributes](#)
 - ◊ [Missing Values](#)
 - ◊ [Data Selection and Sampling](#)
- **Classification**
 - ◊ [Learners and Classifiers](#)
 - ◊ [Probabilistic Classification](#)
 - ◊ [Cross-Validation](#)
 - ◊ [Handful of Classifiers](#)
- **Regression**
 - ◊ [Handful of Regressors](#)
 - ◊ [Cross Validation](#)

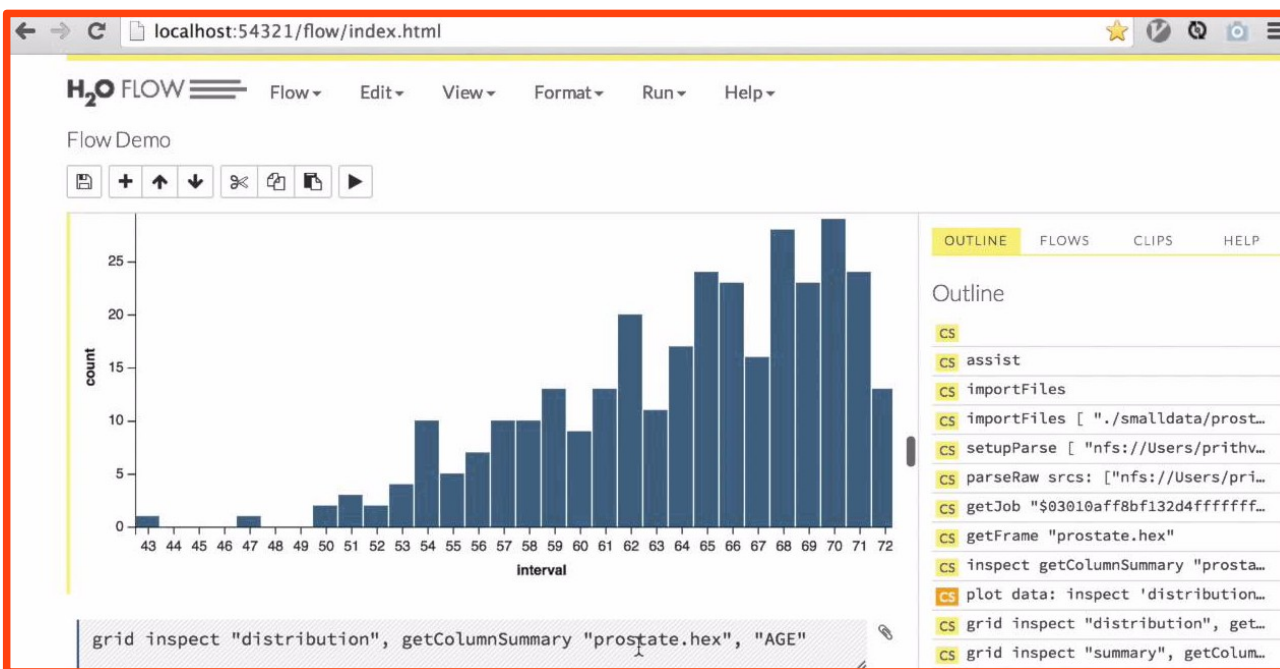
Reference

Available classes and methods.

- **Data model (data)**
 - ◊ [Data Storage \(storage\)](#)
 - ◊ [Data Table \(table\)](#)
 - ◊ [SQL table \(data.sql\)](#)
 - ◊ [Domain description \(domain\)](#)
 - ◊ [Variable Descriptors \(variable\)](#)
 - ◊ [Values \(value\)](#)
 - ◊ [Data Instance \(instance\)](#)
 - ◊ [Data Filters \(filter\)](#)
 - ◊ [Loading and saving data \(io\)](#)
- **Data Preprocessing (preprocess)**
 - ◊ [Impute](#)
 - ◊ [Discretization](#)
 - ◊ [Continuization](#)
 - ◊ [Normalization](#)
 - ◊ [Randomization](#)
 - ◊ [Remove](#)
 - ◊ [Feature selection](#)
 - ◊ [Preprocessors](#)
- **Classification (classification)**
 - ◊ [Logistic Regression](#)
 - ◊ [Random Forest](#)
 - ◊ [Simple Random Forest](#)
 - ◊ [Softmax Regression](#)
 - ◊ [k-Nearest Neighbors](#)
 - ◊ [Naive Bayes](#)

2. Python para Mineração de Dados

- H2O



Unsupervised Learning

Generalized Low Rank Models (GLRM)	Tutorial	Reference
K-Means Clustering	Tutorial	Reference
Principal Components Analysis (PCA)	Tutorial	Reference

Supervised Learning

Generalized Linear Modeling (GLM)	Tutorial	Booklet	Reference	Tuning
Gradient Boosting Machine (GBM)	Tutorial	Booklet	Reference	Tuning
Deep Learning	Tutorial	Booklet	Reference	Tuning
Distributed Random Forest	Tutorial	Booklet	Reference	Tuning
Naive Bayes	Tutorial	Booklet	Reference	Tuning
Stacked Ensembles	Tutorial	Booklet	Reference	Tuning
XGBoost	Tutorial	Booklet	Reference	Tuning

2. Python para Mineração de Dados

- H2O

- > Suporte a processamento distribuído
- > Suporte a aceleração com GPU
- > Interfaces para Python (H2O-Python) e R (H2O-R)

```
import h2o
import imp
from h2o.estimators.kmeans import H2OKMeansEstimator
```

```
# Start a local instance of the H2O engine.
h2o.init();
```

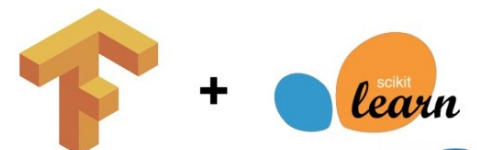
```
iris = h2o.import_file(path="https://github.com/h2oai/h2o-3/raw/master/h2o-r/h2o-package/inst/extdata/iris_wheader.csv")
```

```
results = [H2OKMeansEstimator(k=clusters, init="Random", seed=2, standardize=True) for clusters
in range(2,13)]
for estimator in results:
    estimator.train(x=iris.col_names[0:-1], training_frame = iris)
```

2. Python para Mineração de Dados

- Tensor Flow

- > Biblioteca de código aberto para Mineração de Dados e Aprendizado de Máquina da Google Brain
- > Foco em redes Neurais
- > Interfaces para Python, C++, Java e Go
- > TensorFlow Lite (mobile)
- > Alguns algoritmos:
 - >> K-means clustering
 - >> Random Forests
 - >> Support Vector Machines
 - >> Gaussian Mixture Model clustering
 - >> Linear/logistic regression
- > Outros algoritmos como contribuição
- > Scikit-learn + TensorFlow = Scikit Flow



3. R para Mineração de Dados

- R para mineração de dados

- > R é uma linguagem de programação GPL e ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos
- > Simples
- > Ambiente integrado de desenvolvimento completo (RStudio)
- > Repositório de bibliotecas unificado, organizado e “intuitivo”: **CRAN** - The **C**omprehensive **R** Archive **N**etwork

3. R para Mineração de Dados

>install.packages("package name")

1

**Pre
Modeling
Stage**

- 1. Data Visualization ggplot2, googleVis
- 2. Data Transformation plyr, data.table
- 3. Missing Value Imputations MissForest, MissMDA
- 4. Outlier Detection Outliers, EVIR
- 5. Feature Selection Features, RRF
- 6. Dimension Reduction FactoMineR, CCP

2

**Modeling
Stage**

- 1. Continuous Regression car, randomforest
- 2. Ordinal Regression RMiner, CoreLearn
- 3. Classification Caret, BigRF
- 4. Clustering CBA, RankCluster
- 5. Time Series forecast, LTSA
- 6. Survival survival, Basta

**Data
Analysis
Useful Libraries
in**



**Post
Modeling
Stage**

3

- 1. General Model Validation LSMeans, Comparison
- 2. Regression Validation RegTest, ACD
- 3. Classification Validation BinomTools, DAIM
- 4. Clustering Validation ClustEval, SigClust
- 5. ROC Analysis PROC, TimeROC



Other Libraries

A. Improve performance Rcpp, parallel

B. Work with web XML, jsonlite, httr

C. Report results shiny, RMarkdown

D. Text Mining tm, twitterR

E. Database sqldf, RODBC, RMongo

F. Miscellaneous swirl, reshape2, qcc



3. R para Mineração de Dados

- Pacotes R para mineração de dados

- > Todos no repositório **CRAN**, bibliotecas para:
 - >> Mineração de Dados, Aprendizado de Máquina e Análise Estatística
 - >> Análise de Cluster e Modelos Finitos
 - >> Análise de Séries Temporais
 - >> Estatística Multivariada
 - >> Análise de Dados Espaciais
- > Pacotes mais utilizados por tarefas:
 - >> **Classificação**
 - >>> Árvore de Decisão: rpart, party
 - >>> Random Forest: randomForest, party
 - >>> SVM: e1071, kernlab
 - >>> Redes Neurais: nnet, neuralnet, RSNNS
 - >>> Avaliação de Performance e Métricas: ROCR

3. R para Mineração de Dados

- Pacotes R para mineração de dados

> Pacotes mais utilizados por tarefas:

>> Clusterização

>>> k-means: kmeans, kmeansruns

>>> k-medoids: pam, pamk

>>> Cluster hierárquico: hclust, agnes, diana

>>> DBSCAN: fpc

>>> BIRCH: birch

>>> Validação de Cluster validation: clv, clValid, NbClust

>> Regras de Associação

>>> Regras de Associação: arules (apriori e eclat)

>>> Padrões Sequenciais: arulesSequence

>>> Visualização de Associações: arulesViz

3. R para Mineração de Dados

- Pacotes R para mineração de dados

> Pacotes mais utilizados por tarefas:

>> Mineração de Texto

>>> Mineração de texto: tm

>>> Modelagem de tópicos: topicmodels, lda

>>> Nuvem de palavras: wordcloud

>>> Acesso a dados do Twitter: twitteR

>> Análise de Séries Temporais

>>> Decomposição de Séries Temporais: decomp, decompose, arima, stl

>>> Previsão de Séries Temporais: forecast

>>> Clusterização de Séries Temporais: TSclust

>>> Deformação de Tempo Dinâmico (DTW): dtw

3. R para Mineração de Dados

- Pacotes R para mineração de dados

> Pacotes mais utilizados por tarefas:

>> **Análise de Redes Sociais**

>>> Pacotes básicos: igraph, sna

>>> Medidas de Centralidade: degree, betweenness, closeness, transitivity

>>> Clusterização: clusters, no.clusters

>>> Cliques: cliques, largest.cliques, maximal.cliques, clique.number

>>> Detecção de comunidades: fastgreedy.community, spinglass.community

>>> Base de Dados de Grafos com Neo4j: RNeo4j

3. R para Mineração de Dados

- Pacotes R para mineração de dados

> Pacotes mais utilizados por tarefas:

>> **Suporte a Big Data**

>>> Hadoop: RHadoop, RHIPE

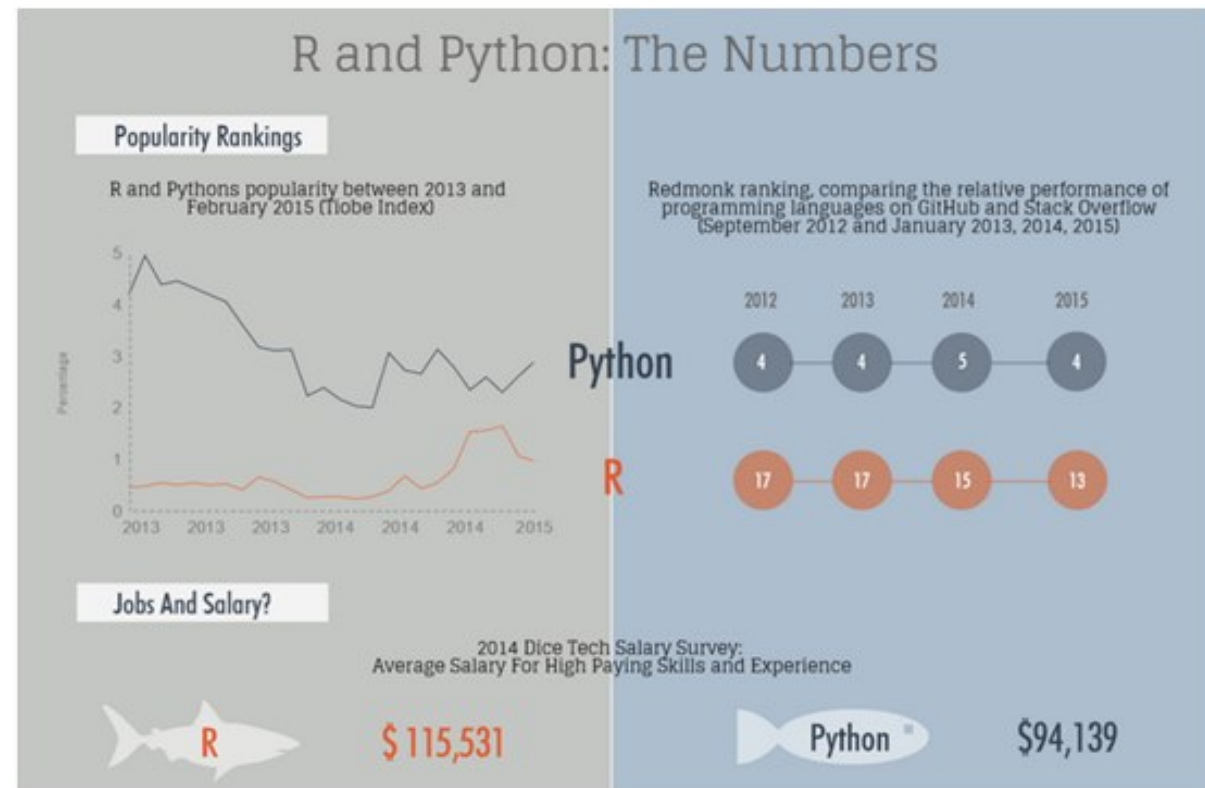
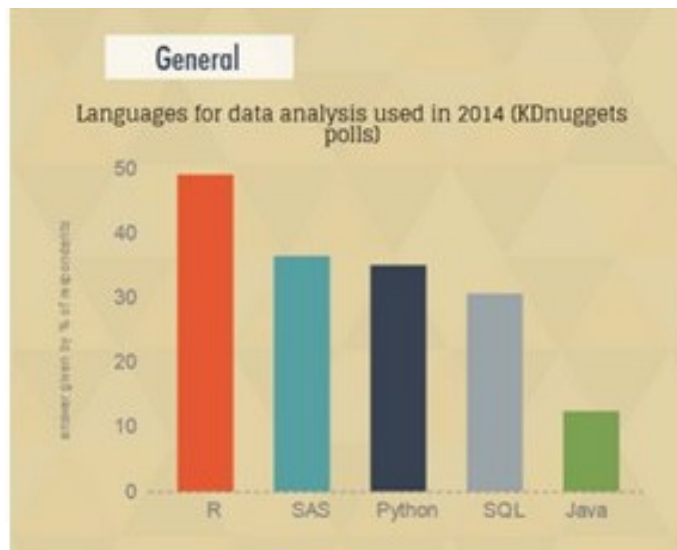
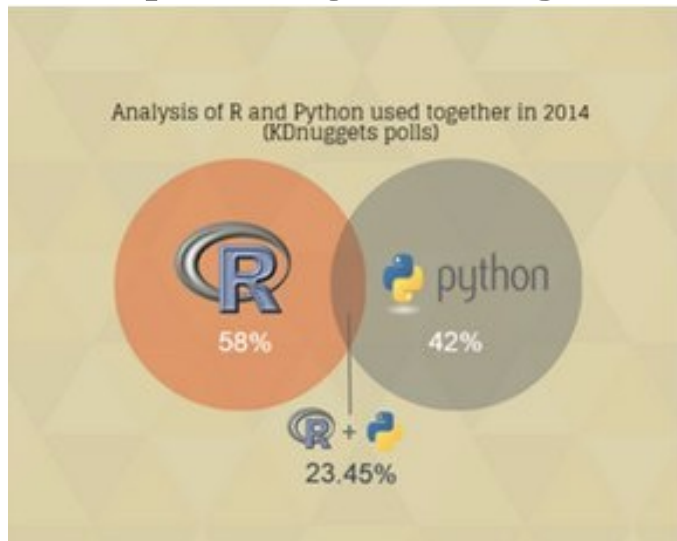
>>> Spark: SparkR

>>> H2O: h2o-r

>>> MongoDB: rmongodb, RMongo

4. Comparação

- Comparação Python e R para mineração de dados



4. Comparação

- Comparação Python e R para mineração de dados

> R

>> Vantagens

- >>> Pacotes de visualização especializados
- >>> Ecosistema R: repositório (CRAN, cran.r-project.org), IDE (Rstudio, rstudio.com) e documentação (rdocumentation.org)
- >>> Linguagem desenvolvida por estatísticos e para estatísticos

>> Desvantagens

- >>> Performance: “R foi feita para facilitar a vida do estatístico, não do computador”
- >>> “A curva de aprendizado do R não é trivial” (palavras de estatísticos que não sabem/gostam de programar, acostumados a ferramentas estatísticas visuais e baseadas em workflows).

4. Comparação

- Comparação Python e R para mineração de dados

> Python

>> Vantagens

>>> Programação/execução interativa com os Notebooks (IPython/Jupyter)

>>> Linguagem de propósito geral/multipropósito

>> Desvantagens

>>> Visualização: escolha complexa, apresentação pouco rica e quantidade de pacotes muito numerosa

>>> Ainda está atrás do R se levarmos em conta a relação quantidade/qualidade/organização dos pacotes, fazendo com que quem já utiliza o R, não o abandone em prol do Python

4. Comparação

- Comparação Python e R para mineração de dados

> R versus Python

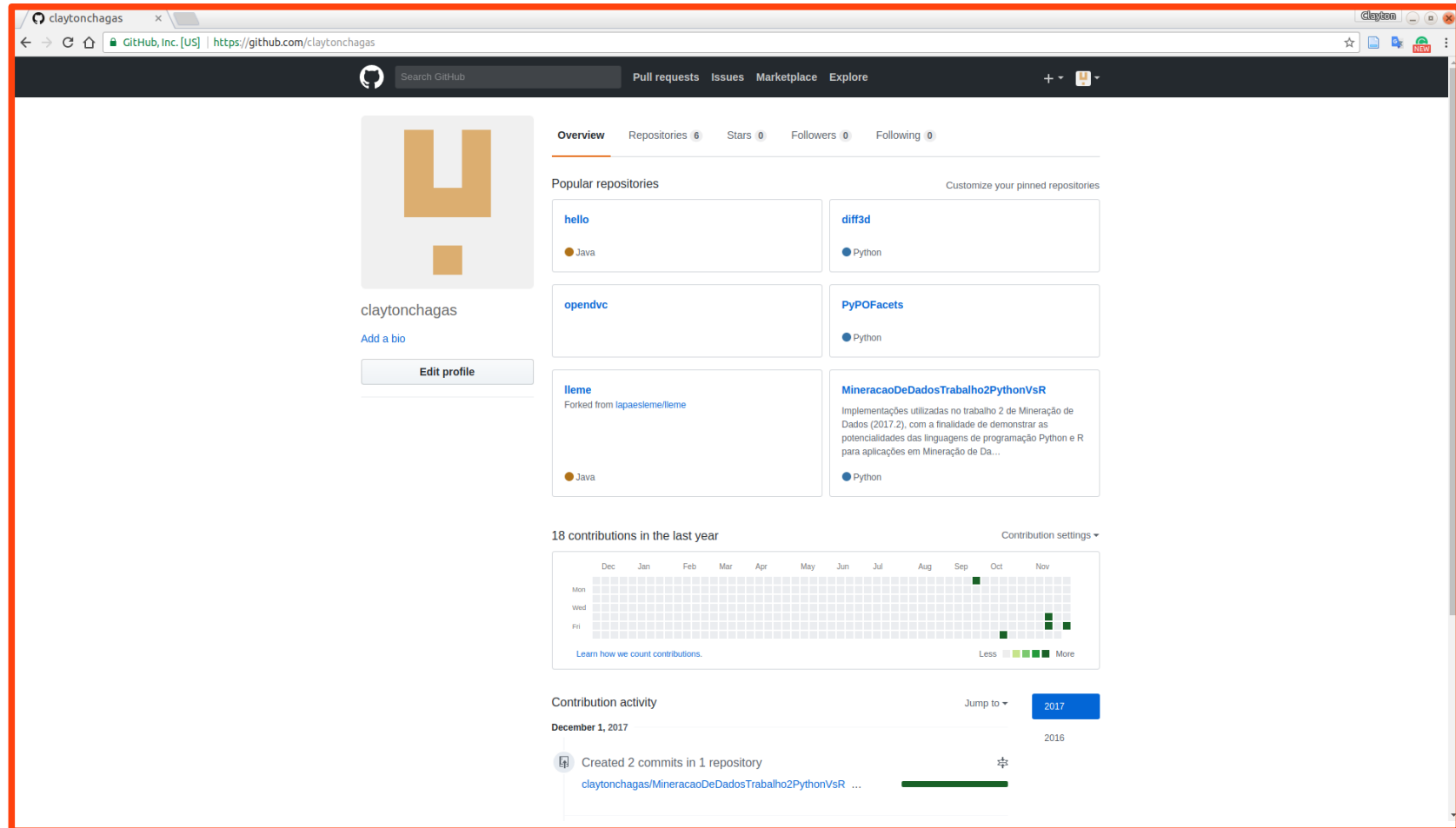
- >> R é mais funcional, Python é mais orientada a objetos
- >> R tem mais funções de mineração embutidas e facilmente encontradas/importadas de seu repositório, Python exige uma análise mais cuidadosa para escolha e tem mais pacotes
- >> R tem melhor suporte à estatística em geral, Python tem mais facilidade para tarefas computacionais

5. Demonstrações

- Algoritmos vistos em sala
 - > Classificação: árvore de decisão, k-NN, Naive Bayes
 - > Clusterização: K-means
- Datasets Mushroom e Iris
- Vídeo: [mineracao.mp4](#)

5. Demonstrações

- Códigos do vídeo e projeto em:
github.com/claytonchagas



5. Demonstrações

- k-NN no Rstudio (biblioteca class: 41 linhas!)

The screenshot displays the RStudio interface. The console on the left contains R code for a k-NN tutorial on the Iris dataset. The environment pane on the right shows the loaded data objects: `ir_test` (45 obs. of 5 variables) and `ir_train` (105 obs. of 5 variables). The bottom pane shows a line plot titled "Error Rate for Iris With Varying K (100 Samples)".

```
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R é um software livre e vem sem GARANTIA ALGUMA.
Você pode redistribuí-lo sob certas circunstâncias.
Digite 'license()' ou 'licence()' para detalhes de distribuição.

R é um projeto colaborativo com muitos contribuidores.
Digite 'contributors()' para obter mais informações e
'citation()' para saber como citar o R ou pacotes do R em publicações.

Digite 'demo()' para demonstrações, 'help()' para o sistema on-line de ajuda,
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu navegador.
Digite 'q()' para sair do R.

> #kNN Tutorial on Iris Data Set####
> library(class) #Has the knn function
> set.seed(4948493) #Set the seed for reproducibility
> #Sample the Iris data set (70% train, 30% test)
> ir_sample<-sample(1:nrow(iris),size=nrow(iris)*.7)
> ir_train<-iris[ir_sample,] #Select the 70% of rows
> ir_test<-iris[-ir_sample,] #Select the 30% of rows
>
> #First Attempt to Determine Right K####
> iris_acc<-numeric() #Holding variable
>
> for(i in 1:50){
+   #Apply knn with k = i
+   predict<-knn(ir_train[,-5],ir_test[,-5],
+               ir_train$Species,k=i)
+   iris_acc<-c(iris_acc,
+               mean(predict==ir_test$Species))
+ }
> #Plot k= 1 through 30
> plot(1-iris_acc,type="l",ylab="Error Rate",
+      xlab="K",main="Error Rate for Iris With Varying K")
>
> #Try many Samples of Iris Data Set to Validate K####
> trial_sum<-numeric(20)
> trial_n<-numeric(20)
> set.seed(6033850)
> for(i in 1:100){
+   ir_sample<-sample(1:nrow(iris),size=nrow(iris)*.7)
+   ir_train<-iris[ir_sample,]
+   ir_test<-iris[-ir_sample,]
+   test_size<-nrow(ir_test)
+   for(j in 1:20){
+     predict<-knn(ir_train[,-5],ir_test[,-5],
+                 ir_train$Species,k=j)
+     trial_sum[j]<-trial_sum[j]+sum(predict==ir_test$Species)
+     trial_n[j]<-trial_n[j]+test_size
+   }
+ }
>
> plot(1-trial_sum / trial_n,type="l",ylab="Error Rate",
+      xlab="K",main="Error Rate for Iris With Varying K (100 Samples)")
```

Environment

Object	Details
<code>ir_test</code>	45 obs. of 5 variables
<code>ir_train</code>	105 obs. of 5 variables

Values

Variable	Values
<code>i</code>	100L
<code>ir_sample</code>	int [1:105] 126 18 63 100 34 30 84 53 116 118 ...
<code>iris_acc</code>	num [1:50] 0.911 0.911 0.933 0.933 0.956 ...
<code>j</code>	20L
<code>predict</code>	Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 ...
<code>test_size</code>	45L
<code>trial_n</code>	num [1:20] 4500 4500 4500 4500 4500 4500 4500 4500 4500 ...
<code>trial_sum</code>	num [1:20] 4310 4283 4318 4308 4333 ...

Files Plots Packages Help Viewer

Error Rate for Iris With Varying K (100 Samples)

6. Conclusão

- R vs Python vs Tools
- R
 - > Computação *standalone* ou análise individualizada de dados
 - > Trabalhos exploratórios e tipos de dados heterogêneos
 - > Cenários instáveis e que precisam ser resolvidos rapidamente
 - > Utilize o RStudio e abuse do CRAN

6. Conclusão

- R vs Python vs Tools
- Python
 - > Integração web ou a outros softwares
 - > Integração a banco de dados em produção
 - > Customizações de algoritmos para produção
 - > Se ambiente a uma boa caixa de ferramentas
 - > Desenvolva utilizando os notebooks (IPython/Jupyter)

6. Conclusão

- R vs Python vs Tools
- DM Tools
 - > Apesar de tudo que foi descrito, vale muito a pena investir no aprendizado de uma ferramenta completa de Mineração de Dados, principalmente com tantas opções gratuitas
 - > Os números apenas traduzem uma questão cultural e/ou comportamental, do ponto de vista de produtividade, do ponto de vista corporativo, utilizar as ferramentas faz mais sentido

7. Referências

- www.kdnuggets.com
- www.analyticsvidhya.com
- www.dataquest.io
- scikit-learn.org
- www.dataconomy.com
- www.guidetodatamining.com
- docs.orange.biolab.si
- www.machinelearningmastery.com
- www.r-project.org
- cran.r-project.org
- www.rdocumentation.org
- www.bioconductor.org
- www.rdatamining.com
- www.datacamp.com