

Data Mining

Disciplina: Machine Learning

Tema da Aula: Algoritmos em Python

Coordenação:

Prof. Dr. Adolpho Walter
Pimazzi Canton

Profa. Dra. Alessandra de
Ávila Montini

Prof. Carlos Eduardo Martins Relvas

Currículo

- Bacharel em Estatística, Universidade de São Paulo.
- Mestre em Estatística, Universidade de São Paulo.
- Itaú, 2010-2015. Principais atividades:
 - Consultoria estatística para várias áreas do banco com foco principal em melhorias no processo de modelagem de risco de crédito.
 - De 2013 a 2015, participação do projeto Big Data do banco usando tecnologia Hadoop e diversas técnicas de machine learning. Desenvolvemos diversos algoritmos em MapReduce usando R e Hadoop streaming, criando uma plataforma de modelagem estatística no Hadoop.
- Nubank, desde 2015. Principais atividades:
 - Equipe de Data Science, responsável por toda a parte de modelagem da empresa, desde modelos de crédito a identificar motivos de atendimento.

Conteúdo da Aula

- Introdução ao Python
- Regressão Linear – Python
- Árvore de decisão – Python
- Random Forest - Python

Python

Python

- Criado em 1990 por Guido van Rossum.
- Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).

Python

- Python 1.0 criado em 1994
- Python 2.0 criado em 2000
- Python 3.0 criado em 2008
- Versão recomendada e mais utilizada é a 2.7 (2010).
- Os básicos da versão 2.7 e 3 são os mesmos (exceto o print). A versão 3 será o futuro da linguagem, mas ainda não possui todos os pacotes disponíveis. Além disso, não possui compatibilidade com os códigos da versão 2.7

Python – Características

- Multi funções (WEB, Gui, Scripting, etc)
- Orientado a objeto
- Intrepretado
- Focado em produtividade (fácil aprendizado) e fácil leitura (manutenção)
- Strongly typed (força erro) e dynamically typed (não precisa definir os tipos de variáveis a priori).

Python – Características

- Multi funções (WEB, Gui, Scripting, etc)
- Orientado a objeto
- Intrepretado
- Focado em produtividade (fácil aprendizado) e fácil leitura (manutenção)
- Strongly typed (força erro) e dynamically typed (não precisa definir os tipos de variáveis a priori).

Python – Características

- Identação obrigatória

```
/* Bogus C code */
```

```
if (foo)
```

```
if (bar)
```

```
baz(foo, bar);
```

```
else
```

```
qux();
```

```
# Python code
```

```
if foo:
```

```
    if bar:
```

```
        baz(foo, bar)
```

```
    else:
```

```
        qux()
```

Python x Perl

- Python é tido como mais fácil de aprender
- Código mais fácil para ler
- Melhor integração com JAVA
- Menos viés do Unix.
- Manutenção mais fácil

Python x Java

- Código de 5 a 10 vezes mais conciso
- Dynamic typing
- Desenvolvimento mais rápido
- Java roda mais rápido

Python - Instalando

- Recomendo o uso do Anaconda

GET SUPERPOWERS WITH ANACONDA

Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R and Scala packages for data science.

Additionally, you'll have access to over 720 packages that can easily be installed with conda, our renowned package, dependency and environment manager, that is included in Anaconda. See the [packages](#) included with Anaconda and the Anaconda [changelog](#)

Which version should I download and install?

With Anaconda you can run multiple versions of Python in isolated environments, so choose the download with the Python version that you use more often, as that will be your default Python version.

If you don't have time or disk space for the entire distribution, try [Miniconda](#) which contains only conda and Python. Then install just the individual packages you want through the conda command.

Python – Básico

- Strings

```
# This is a string
name = "Nowell Strite (that\"s me)"

# This is also a string
home = 'Huntington, VT'

# This is a multi-line string
sites = '''You can find me online
on sites like GitHub and Twitter.'''

# This is also a multi-line string
bio = """If you don't find me online
you can find me outside."""
```

Python – Básico

- Números

```
# Integers Numbers
```

```
year = 2010
```

```
year = int("2010")
```

```
# Floating Point Numbers
```

```
pi = 3.14159265
```

```
pi = float("3.14159265")
```

```
# Fixed Point Numbers
```

```
from decimal import Decimal
```

```
price = Decimal("0.02")
```

Python – Básico

- Listas

```
# Lists can be heterogeneous  
favorites = []
```

```
# Appending  
favorites.append(42)
```

```
# Extending  
favorites.extend(["Python", True])
```

```
# Equivalent to  
favorites = [42, "Python", True]
```


Python – Básico

- Listas

```
numbers = [1, 2, 3, 4, 5]
```

```
len(numbers)  
# 5
```

```
numbers[0]  
# 1
```

```
numbers[0:2]  
# [1, 2]
```

```
numbers[2:]  
# [3, 4, 5]
```

Python – Básico

- Dicionários

```
person = {}
```

```
# Set by key / Get by key  
person['name'] = 'Nowell Strite'
```

```
# Update  
person.update({  
    'favorites': [42, 'food'],  
    'gender': 'male',  
})
```

```
# Any immutable object can be a dictionary key  
person[42] = 'favorite number'  
person[(44.47, -73.21)] = 'coordinates'
```

Python – Básico

- Dicionários

```
person = {'name': 'Nowell', 'gender': 'Male'}

person['name']
person.get('name', 'Anonymous')
# 'Nowell Strite'

person.keys()
# ['name', 'gender']

person.values()
# ['Nowell', 'Male']

person.items()
# [['name', 'Nowell'], ['gender', 'Male']]
```

Python – Básico

- Booleanos

```
# This is a boolean  
is_python = True
```

```
# Everything in Python can be cast to boolean  
is_python = bool( Python Programming Language )
```

```
# All of these things are equivalent to False  
these_are_false = False or 0 or "" or {} or []  
or None
```

```
# Most everything else is equivalent to True  
these_are_true = True and 1 and "Text" and  
{ 'a': 'b' } and [ 'c', 'd' ]
```

Python – Básico

- Matemática

```
a = 10           # 10
```

```
a += 1           # 11
```

```
a -= 1           # 10
```

```
b = a + 1        # 11
```

```
c = a - 1        # 9
```

```
d = a * 2         # 20
```

```
e = a / 2         # 5
```

```
f = a % 3         # 1
```

```
g = a ** 2        # 100
```

Python – Básico

- Manipulação Strings

```
animals = "Cats " + "Dogs "  
animals += "Rabbits"  
# Cats Dogs Rabbits
```

```
fruit = ', '.join(['Apple', 'Banana', 'Orange'])  
# Apple, Banana, Orange
```

```
date = '%s %d %d' % ('Sept', 11, 2010)  
# Sept 11 2010
```

```
name = '%(first)s %(last)s' % {  
    'first': 'Nowell',  
    'last': 'Strite'}  
# Nowell Strite
```

Python – Básico

- Lógica

```
# Logical And
```

```
a and b
```

```
# Logical Or
```

```
a or b
```

```
# Logical Negation
```

```
not a
```

```
# Compound
```

```
(a and not (b or c))
```

Python – Básico

- Comparação

```
# Identity  
1 is 1 == True
```

```
# Non Identity  
1 is not '1' == True
```

```
# Example  
bool(1) == True  
bool(True) == True
```

```
1 and True == True  
1 is True == False
```


Python – Básico

- Condicional

```
grade = 82
if grade >= 90:
    if grade == 100:
        print 'A+'
    else:
        print "A"
elif grade >= 80:
    print "B"
elif grade >= 70:
    print "C"
else:
    print "F"
```

Python – Básico

- Looping

```
for x in range(10): #0-9
    print x
```

```
fruits = ['Apple', 'Orange']
```

```
for fruit in fruits:
    print fruit
```

```
x = 0
while x < 100:
    print x
    x += 1
```

Python – Básico

- List Comprehensions

```
odds = [ x for x in range(50) if x % 2 ]
```

```
odds = []  
for x in range(50):  
    if x % 2:  
        odds.append(x)
```

Python – Básico

- Funções

```
def my_function():  
    """Function Documentation"""  
    print "Hello World"
```

```
# Positional  
def add(x, y):  
    return x + y
```

```
# Keyword  
def shout(phrase='Yipee!'):  
    print phrase
```

```
# Positional + Keyword  
def echo(text, prefix=''):  
    print '%s%s' % (prefix, text)
```

Python – Básico

- Argumentos arbitrários

```
def some_method(*args, **kwargs):  
    for arg in args:  
        print arg  
  
    for key, value in kwargs.items():  
        print key  
  
some_method(1, 2, 3, name='Numbers')
```

Python – Básico

- Imports

```
# Imports the datetime module into the
# current namespace
import datetime
datetime.date.today()
datetime.timedelta(days=1)

# Imports datetime and adds date and
# timedelta into the current namespace
from datetime import date, timedelta
date.today()
timedelta(days=1)
```

Python – Básico

- Imports

```
# Renaming imports
from datetime import date
from my_module import date as my_date

# This is usually considered a big No-No
from datetime import *
```

Python – Pacotes

- Python, assim como o R, tem vários pacotes para várias tarefas distintas. Para Data Science, podemos destacar:
 - Pandas
 - Numpy
 - Sklearn
 - Matplotlib (gráficos)

Python – Pandas

- **Panel Data Structures**
- Ideia principal: indexar estrutura de dados capaz de armazenar dados heterogêneos.
- Funcionalidade parecida com SQL (joining / merging, GroupBy).
- Capaz de lidar com dados missing.
- Preparação de dados (limpeza e merge dados de diferentes fontes como CSV, Excel, etc).

Python – Pandas

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pd.set_option('max_columns', 50)
%matplotlib inline
```

```
# create a Series with an arbitrary list
s = pd.Series([7, 'Heisenberg', 3.14, -1789710578, 'Happy Eating!'])
s
```

```
0          7
1    Heisenberg
2         3.14
3   -1789710578
4    Happy Eating!
dtype: object
```

Python – Pandas

```
d = {'Chicago': 1000, 'New York': 1300, 'Portland': 900, 'San Francisco': 1100,  
     'Austin': 450, 'Boston': None}  
cities = pd.Series(d)  
cities
```

Austin	450
Boston	NaN
Chicago	1000
New York	1300
Portland	900
San Francisco	1100
dtype:	float64

Python – Pandas

```
cities['Chicago']
```

```
1000.0
```

```
cities[['Chicago', 'Portland', 'San Francisco']]
```

```
Chicago      1000  
Portland      900  
San Francisco 1100  
dtype: float64
```

```
cities[cities < 1000]
```

```
Austin      450  
Portland     900  
dtype: float64
```

Python – Pandas

```
# divide city values by 3  
cities / 3
```

Austin	250.000000
Boston	NaN
Chicago	466.666667
New York	433.333333
Portland	250.000000
San Francisco	366.666667

dtype: float64

```
# square city values  
np.square(cities)
```

Austin	562500
Boston	NaN
Chicago	1960000
New York	1690000
Portland	562500
San Francisco	1210000

dtype: float64

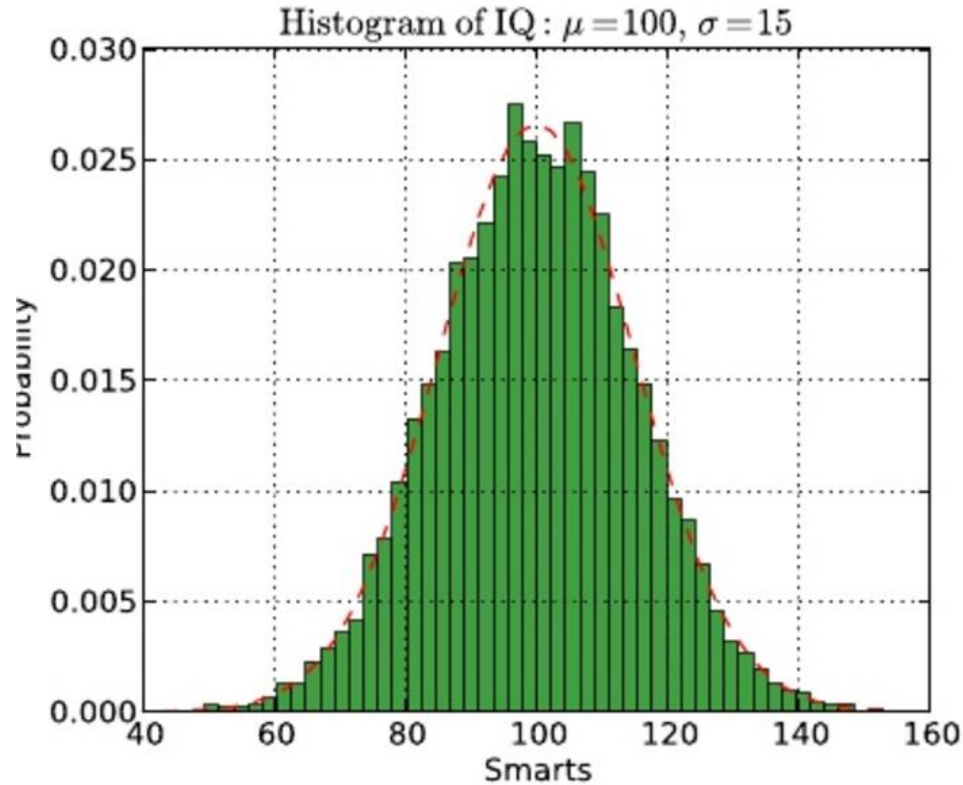
Python – NumPy

- Pacote padrão para computação numérica e científica em Python.
- Vetor N-dimensional sofisticado.
- Várias funções numéricas que servem como bas para vários pacotes.
- Ferramentas para integração com C/C++ e Fortran.
- Útil para algébra linear, transformada de Fourier e geração de números aleatórios.

Python – Matplotlib

- Pacote para construção de diversos gráficos em 2D como gráficos de dispersão, histogramas e boxplots.
- Similar com as ferramentas de visualização disponíveis no R base
 - Gráficos do R são muito mais fáceis de serem construídos, requerendo menos código.
 - Gráficos do Matplotlib tendem a ser mais bonitos.

Python – Matplotlib



Python – SkLearn (SciKit Learn)

- Pacote que possibilita o treino de diferentes técnicas de estatística e machine learning.
- Similar ao pacote caret do R.

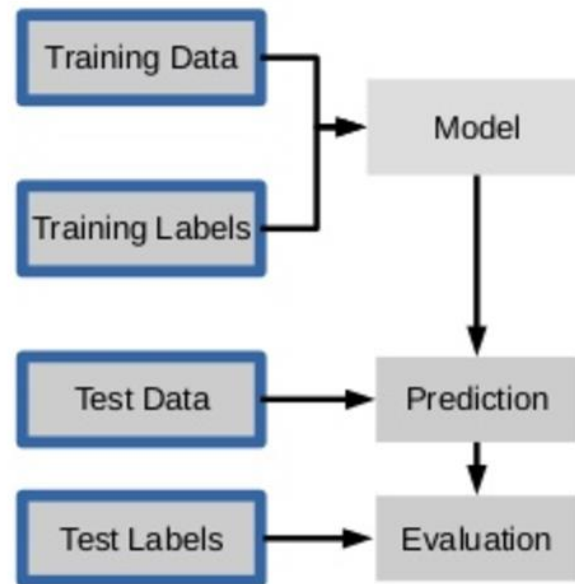
Python – SkLearn (modelos supervisionados)

```
clf = RandomForestClassifier()
```

```
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
```

```
clf.score(X_test, y_test)
```

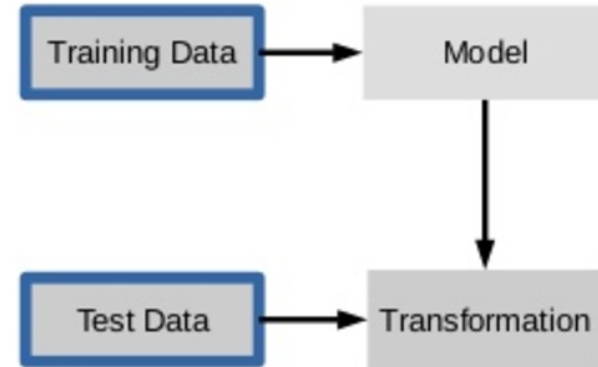


Python – SkLearn (modelos não supervisionados)

```
pca = PCA(n_components=3)
```

```
pca.fit(X_train)
```

```
X_new = pca.transform(X_test)
```



Python – SkLearn

```
estimator.fit(X, [y])
```

```
estimator.predict
```

```
estimator.transform
```

Classification

Preprocessing

Regression

Dimensionality reduction

Clustering

Feature selection

Feature extraction

Python – SkLearn (cross validation)

```
from sklearn.cross_validation import cross_val_score

scores = cross_val_score(SVC(), X, y, cv=5)
print(scores)

>> [ 0.92  1.    1.    1.    1. ]

cv_ss = ShuffleSplit(len(X_train), test_size=.3,
                    n_iter=10)
scores_shuffle_split = cross_val_score(SVC(), X, y,
                                       cv=cv_ss)

cv_labels = LeaveOneLabelOut(labels)
scores_pout = cross_val_score(SVC(), X, y, cv=cv_labels)
```

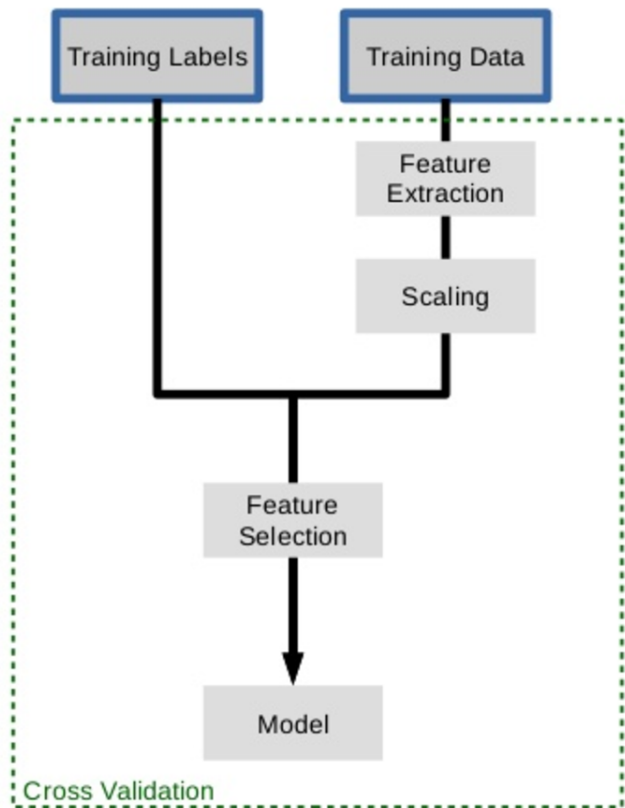
Python – SkLearn (cross validation)

```
from sklearn.grid_search import GridSearchCV
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

param_grid = {'C': 10. ** np.arange(-3, 3),
              'gamma': 10. ** np.arange(-3, 3)}
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
grid.predict(X_test)
grid.score(X_test, y_test)
```

Python – SkLearn (pipelines)



```
from sklearn.pipeline import make_pipeline

pipe = make_pipeline(StandardScaler(), SVC())
pipe.fit(X_train, y_train)
pipe.predict(X_test)
```

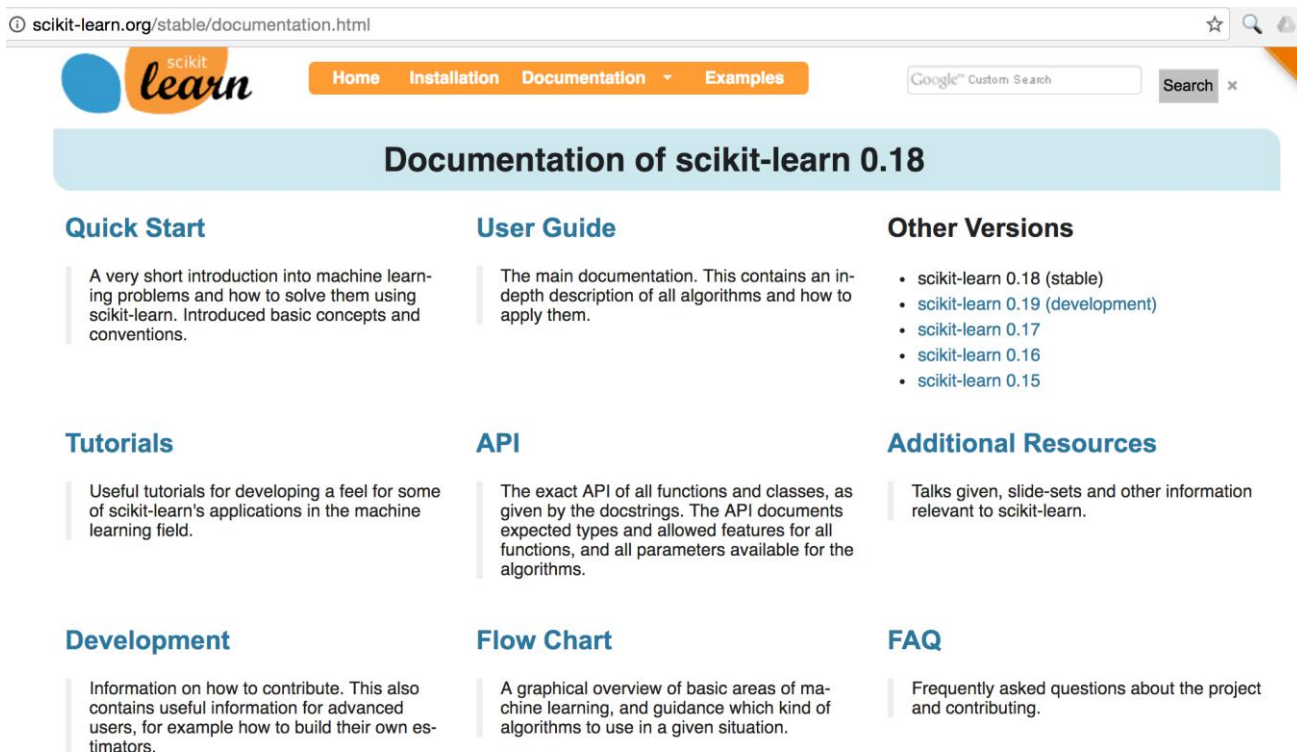
Python – SkLearn (GridSearch + pipelines)

```
param_grid = {'svc__C': 10. ** np.arange(-3, 3),  
              'svc__gamma': 10. ** np.arange(-3, 3)}  
  
scaler_pipe = make_pipeline(StandardScaler(), SVC())  
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)  
grid.fit(X_train, y_train)
```


Python – SkLearn (Métricas)

```
print(SCORERS.keys())  
  
>> ['adjusted_rand_score',  
      'f1',  
      'mean_absolute_error',  
      'r2',  
      'recall',  
      'median_absolute_error',  
      'precision',  
      'log_loss',  
      'mean_squared_error',  
      'roc_auc',  
      'average_precision',  
      'accuracy']
```

Python – SkLearn (Documentação)



The screenshot shows the scikit-learn documentation website. The browser address bar displays 'scikit-learn.org/stable/documentation.html'. The website header includes the 'scikit-learn' logo, navigation links for 'Home', 'Installation', 'Documentation', and 'Examples', and a Google Custom Search bar. A light blue banner below the header reads 'Documentation of scikit-learn 0.18'. The main content area is organized into a 3x3 grid of links, each with a title and a brief description.

Quick Start	User Guide	Other Versions
A very short introduction into machine learning problems and how to solve them using scikit-learn. Introduced basic concepts and conventions.	The main documentation. This contains an in-depth description of all algorithms and how to apply them.	<ul style="list-style-type: none">• scikit-learn 0.18 (stable)• scikit-learn 0.19 (development)• scikit-learn 0.17• scikit-learn 0.16• scikit-learn 0.15
Tutorials	API	Additional Resources
Useful tutorials for developing a feel for some of scikit-learn's applications in the machine learning field.	The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.	Talks given, slide-sets and other information relevant to scikit-learn.
Development	Flow Chart	FAQ
Information on how to contribute. This also contains useful information for advanced users, for example how to build their own estimators.	A graphical overview of basic areas of machine learning, and guidance which kind of algorithms to use in a given situation.	Frequently asked questions about the project and contributing.

Exercícios

1. Calcule $\sum_{i=1}^{100.000} \sin(i)^2$ utilizando um loop e de forma vetorial.
2. Use a base de dados “Gastos_Cartao”
 - a.) Selecione as colunas ‘Idade’, ‘Renda’ e ‘Gastos Cartao’. Calcule a média e o desvio padrão de cada uma destas colunas.
 - b.) Filtre apenas segmento ‘B’, e refaça o exercício a.)

Exercícios

3. Escreva uma função que receba dois argumentos (x e n), em que x é um número qualquer e n é um número estritamente positivo. A função deve retornar o valor de:

$$1 + \frac{x}{1} + \frac{x^2}{2} + \dots + \frac{x^n}{n}$$

4. Seja um anglo α é dado por um número positivo em graus. Se $0 \leq \alpha < 90$, então é o quadrante 1. Se $90 \leq \alpha < 180$, então é quadrante 2. Se $180 \leq \alpha < 270$, então é quadrante 3. Se $270 \leq \alpha < 360$, então é quadrante 4. Se $360 \leq \alpha < 450$, então é quadrante 1 e assim por diante. Escreva um função que dado um valor alpha retorna o quadrante correspondente.

ML models - Laboratório – Regressão

Base simulada com 150 observações e 5 variáveis.

- Gastos no cartão em reais
- Idade
- Renda
- Pagamento de impostos
- Segmento

Objetivo:

Prever os Gastos no cartão com base nas outras informações.

```
> head(dados)
```

	Gastos_Cartao	Idade	Renda	Impostos	Segmento
1	510	35	1120	60	C
2	490	30	1120	60	C
3	470	32	1040	60	C
4	460	31	1200	60	C
5	500	36	1120	60	C
6	540	39	1360	120	C

Laboratório – Regressão

Notebook de análise: Gastos_cartao.ipynb

Laboratório – Classificação

Laboratório R - Base de Spam

- Base com 4.601 e-mails. Porcentual em que 54 palavras ou pontuações aparecem em cada e-mail. Além disso, temos o tamanho médio das palavras, tamanho da maior palavra e quantidade de palavras.

Objetivo:

Criar um detector automático de SPAM que verificará cada novo e-mail.

- Disponível em: <https://archive.ics.uci.edu/ml/datasets/Spambase>

Laboratório – Classificação

Notebook de análise: Spam.ipynb

Laboratório – Exercício

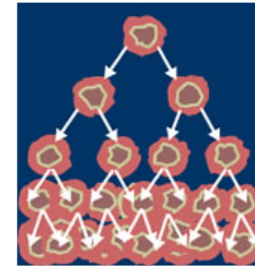
<http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>



Breast Cancer Wisconsin (Diagnostic) Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Diagnostic Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	Number of Instances:	569	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	32	Date Donated	1995-11-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	390512

Laboratório – Exercício

Base de dados (“cancer.data”) com 699 observações e 10 variáveis de pacientes com tumores. O objetivo é detectar com base em algumas informações dos tumores se é benigno ou maligno.

- Variáveis:
- 1. Sample code number id number
 - 2. Clump Thickness 1 - 10
 - 3. Uniformity of Cell Size 1 - 10
 - 4. Uniformity of Cell Shape 1 – 10
 - 5. Marginal Adhesion 1 - 10
 - 6. Single Epithelial Cell Size 1 - 10
 - 7. Bare Nuclei 1 - 10
 - 8. Bland Chromatin 1 - 10
 - 9. Normal Nucleoli 1 - 10
 - 10. Mitoses 1 - 10
 - 11. Class: (2 for benign, 4 for malignant)

Laboratório – Exercício

- 1.) Crie bases de desenvolvimento (70%) e teste (30%). Utilize seed de 42.
- 2.) Cheque e corrija possíveis problemas de missing.
- 3.) Construa a regressão logística, árvore de decisão e random forest.
- 4.) Otimize os hiper-parâmetros.
- 5.) Avalie os resultados.

Referências Bibliográficas

- Hastie, T., Tibshirani, R. & Friedman, J.H. (2001) “The Elements of Statistical Learning”
- Bishop, C.M. (2007) “Pattern Recognition and Machine Learning”
- Mitchell, T.M. (1997) “Machine Learning”
- Abu-Mostafa, Y., Magdon-Ismail, M., Lin, H.T (2012) “Learning from data”
- Theodoridis, S., Koutroumbas, K., (2008) “Pattern Recognition”
- Kuhn, M., Johnson, K., (2013) “Applied Predictive Modeling”