Data mining

Técnicas e ferramentas

Apresentado por Marcelo ML Lima

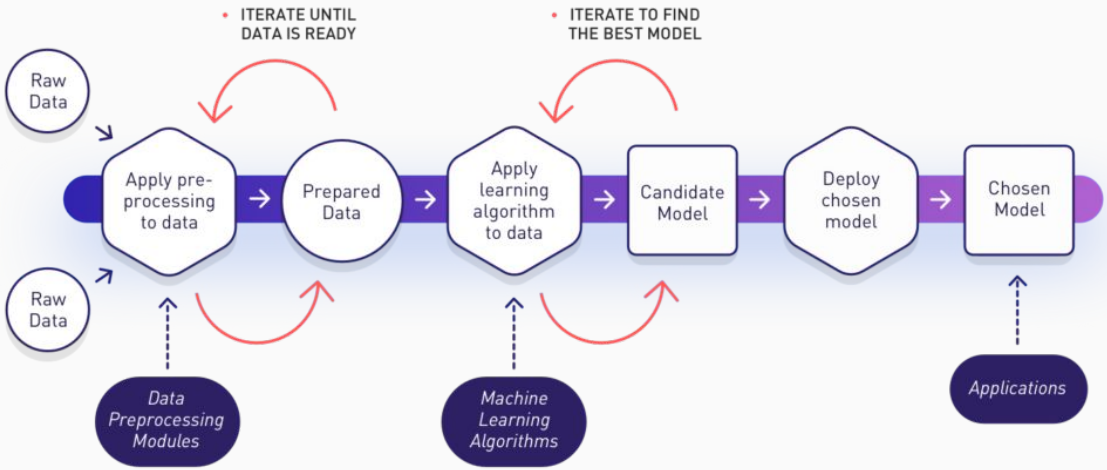
Na aula de hoje

Tópicos da aula

- **Ferramentas no Life Cycle**
- Aquisição de dados
- Pre Processamento
- Modelos de IA
- MLOps - Conceito e Ferramentas
- **Exemplo em problema**

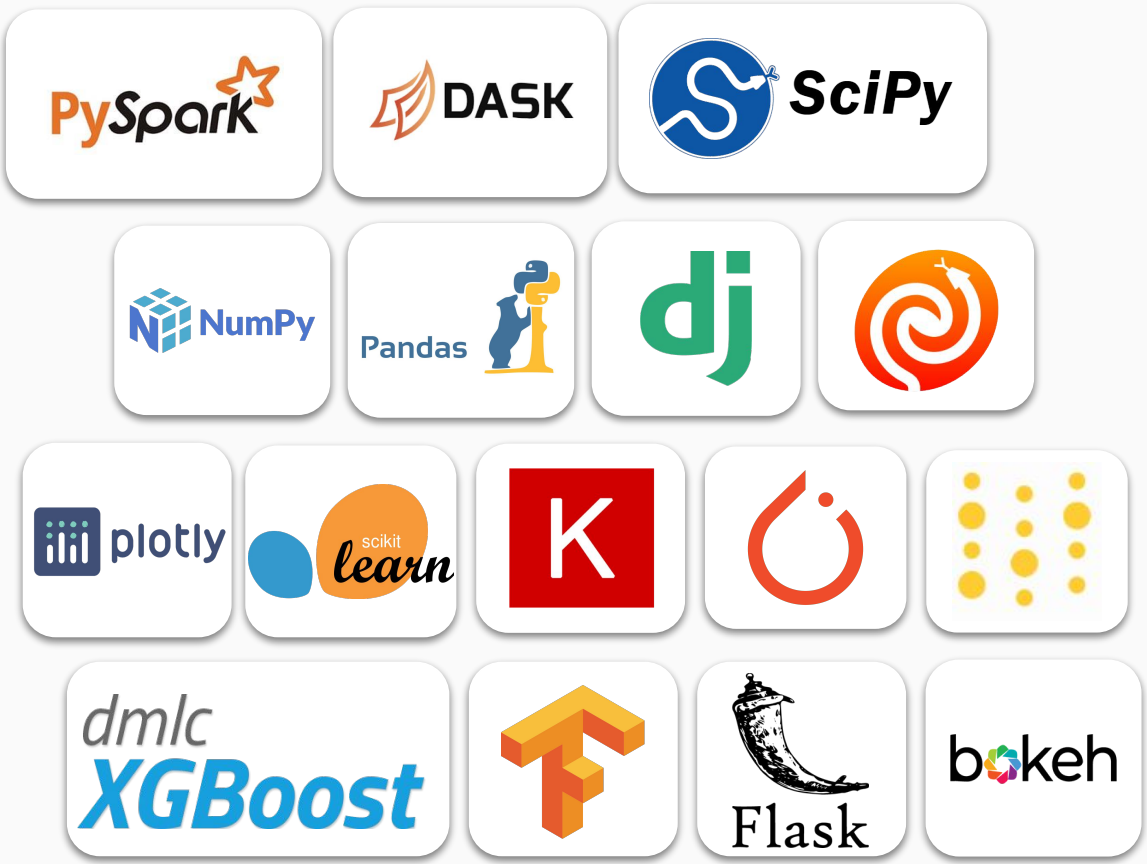
Da aula anterior...

Relembrando a estrutura do projeto



Conhecendo as ferramentas

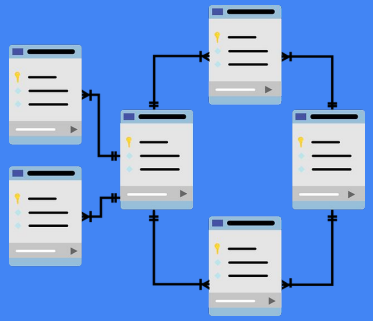
Das seguintes ferramentas, onde cada uma se encaixa dentro do Life Cycle de um projeto de Machine Learning?



Ferramentas

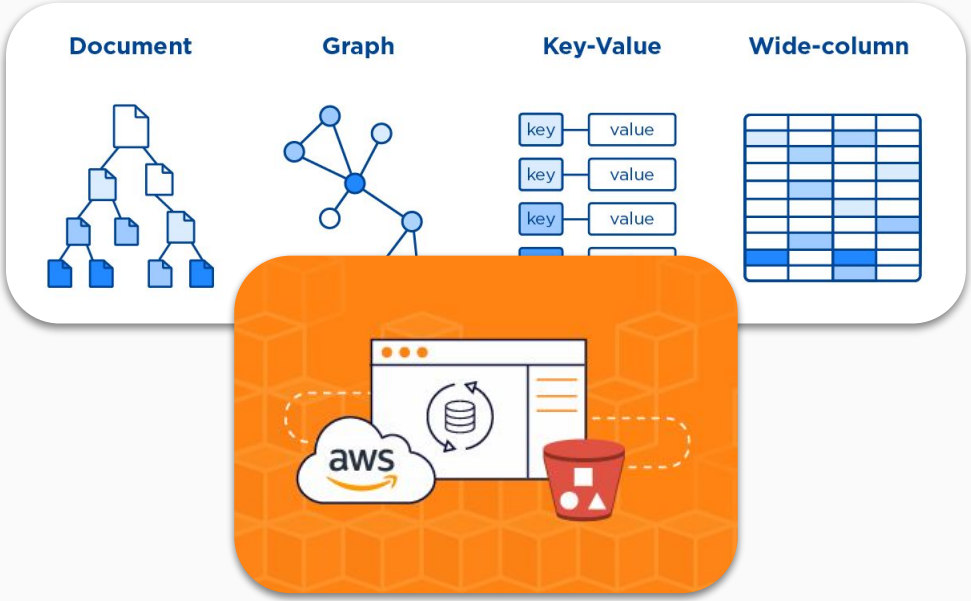
Nessa aula vamos discutir as melhores ferramentas, do ponto de vista de performance e usabilidade no mercado, para cada etapa do Life Cycle de um projeto de Machine Learning. A ideia é construir um arcabouço de ferramentas práticas, impactantes e performáticas para soluções de Machine Learning.

Aquisição de dados



Quais os tipos de fontes de dados que podem existir em nosso projeto?

- Banco de dados relacionais;
- Banco de dados não relacionais;
- Buckets - Data Blobs;
- Web Scraping;
- APIs;
- Arquivos locais;
- ...



Tipo de dados

Essas são as bibliotecas que possivelmente serão utilizadas para o processamento de cada tipo de dado para o seu problema de machine learning.

Note que tais bibliotecas não são mandatórias nem excludentes, desta forma, podem existir problemas que você deverá utilizar mais que uma biblioteca, ou nem uma das bibliotecas apresentadas.

Tabelas

Imagens

Series Temporais

Requests


```python
import requests

# Making a GET request
r = requests.get('https://api.github.com/events', params = {'key1': 'value1', 'key2': ['value2', 'value3']})

print(r.content()) # Return the binary content: r'{'content': Example of content}'
print(r.json()) # Return the content as json: {'content': Example of content}

# Making a POST request
r = requests.post('https://httpbin.org/post', data = {'key':'value'})

# Other examples of requests
r = requests.put('https://httpbin.org/put', data = {'key':'value'})
r = requests.delete('https://httpbin.org/delete')
r = requests.head('https://httpbin.org/get')
r = requests.options('https://httpbin.org/get')
```

Firebase


```python
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db

# Fetch the service account key JSON file contents
cred = credentials.Certificate('path/to/serviceAccountKey.json')

# Initialize the app with a service account, granting admin privileges
firebase_admin.initialize_app(cred, {
    'databaseURL': 'https://databaseName.
})

# As an admin, the app has access to read
ref = db.reference('restricted_access/sec
print(ref.get())
```

```
{
"rules": {
    "public_resource": {
    ".read": true,
    ".write": true
    },
    "some_resource": {
    ".read": "auth.uid === 'my-service-worker'",
    ".write": false
    },
    "another_resource": {
    ".read": "auth.uid === 'my-service-worker'",
    ".write": "auth.uid === 'my-service-worker'"
    }
}
}
```

```python
users_ref = ref.child('users')

# Set all the Users Reference Structure
users_ref.set({
    'alanisawesome': {
        'date_of_birth': 'June 23, 1912',
        'full_name': 'Alan Turing'
    },
    'gracehop': {
        'date_of_birth': 'December 9, 1906',
        'full_name': 'Grace Hopper'
    }
})

# Set particular Users inside Users Structure
users_ref.child('alanisawesome').set({
    'date_of_birth': 'June 23, 1912',
    'full_name': 'Alan Turing'
})
users_ref.child('gracehop').set({
    'date_of_birth': 'December 9, 1906',
    'full_name': 'Grace Hopper'
})
```

boto3


```python
import boto3

s3 = boto3.resource('s3')

# Print out bucket names
for bucket in s3.buckets.all():
    print(bucket.name)

# Upload a new file
data = open('test.jpg', 'rb')
s3.Bucket('my-bucket').put_object(Key='test.jpg', Body=data)

# Download file object
with open('FILE_NAME', 'wb') as f:
    s3.download_fileobj('BUCKET_NAME', 'OBJECT_NAME', f)
```

```python
# Reading CSV from AWS Bucket
import pandas as pd
import boto3

# get your credentials from environment variables
aws_id = os.environ['AWS_ID']
aws_secret = os.environ['AWS_SECRET']

client = boto3.client('s3', aws_access_key_id=aws_id,
        aws_secret_access_key=aws_secret)

bucket_name = 'my_bucket'

object_key = 'my_file.csv'
csv_obj = client.get_object(Bucket=bucket_name, Key=object_key)
body = csv_obj['Body']
csv_string = body.read().decode('utf-8')

df = pd.read_csv(StringIO(csv_string))
```

Google Cloud Storage


```python
# Imports the Google Cloud client library
from google.cloud import storage

# Instantiates a client
storage_client = storage.Client()

# The name for the new bucket
bucket_name = "my-new-bucket"

# Creates the new bucket
bucket = storage_client.create_bucket(bucket_name)

print("Bucket {} created.".format(bucket.name))
```

```python
from google.cloud import storage

# Get Storage Client
client = storage.Client()

# Get the Bucket Client
bucket = client.get_bucket('my-bucket-name')
# Get the blob file from Bucket
blob = storage.Blob('path/to/blob', bucket)

# 1 - First Way to download a file from bucket
with open('file-to-download-to') as file_obj:
    client.download_blob_to_file(blob, file_obj)  # API request.

# 2 - Second Way to download a file from bucket
with open('file-to-download-to') as file_obj:
    client.download_blob_to_file(
        'gs://bucket_name/path/to/blob', file_obj)
```

Beautiful Soup

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')

print(soup.prettify())
```

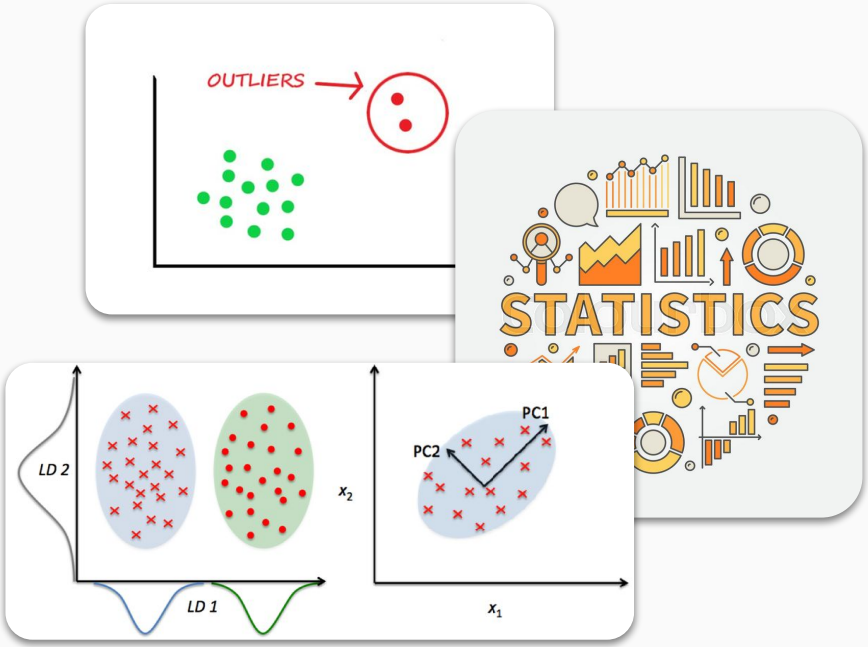
```
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three little sisters; and their names were
   <a class="sister" href="http://example.com/elsie" id="link1">
    Elsie
   </a>
   ,
   <a class="sister" href="http://example.com/lacie" id="link2">
    Lacie
   </a>
   and
   <a class="sister" href="http://example.com/tillie" id="link3">
    Tillie
   </a>
   ; and they lived at the bottom of a well.
  </p>
  <p class="story">
   ...
  </p>
 </body>
</html>
```

```
soup.title

soup.title.name

soup.title.string

soup.title.parent.name

soup.p

soup.p['class']

soup.a

soup.find_all('a')

soup.find(id="link3")
```

Pre processamento

Quais os tipos de processamentos que podem existir em nosso projeto?

- Filtragem dos dados;
- Manipulação dos dados;
- Transformação dos dados;
- Extração de features;
- Redução de dimensionalidade;
- Normalização/Standarization;
- Visualização de dados.



Pandas

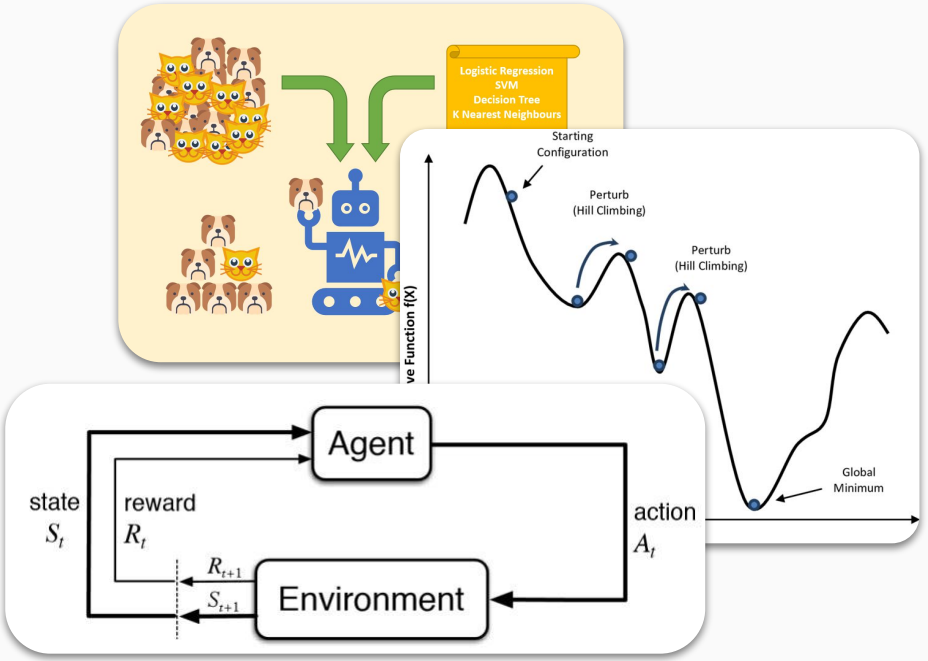

```python
import pandas as pd
import pyodbc

conn = pyodbc.connect(
    r'Driver={Microsoft Access Driver (*.mdb, *.accdb)};DBQ=C:\Users\Ron\Desktop\testdb.accdb;'
)

SQL_Query = pd.read_sql_query('''
    select
        product_name,
        product_price_per_unit,
        units_ordered,
        ((units_ordered) * (product_price_per_unit)) AS revenue
    from tracking_sales''', conn)

df = pd.DataFrame(SQL_Query, columns=['field1','field2',...])
```

```python
from google.oauth2 import service_account
import pandas_gbq

credentials = service_account.Credentials.from_service_account_file(
    'path/to/key.json',
)
df = pandas_gbq.read_gbq(sql, project_id="YOUR-PROJECT-ID", credentials=credentials)
```

Modelos

Quais os tipos de problemas de ML que podem existir em nosso projeto?

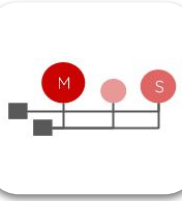
- Clusterização
- Classificação - Tabelas
- Regressão
- Classificação - Imagens
- Séries Temporais
- Otimização
- Aprendizagem por reforço



Tipo de problemas

Essas são as bibliotecas que possivelmente serão utilizadas para resolver o tipo de problema apresentado. Mas note que diversos problemas podem ser consolidados em estruturas de Regressão e/ou Otimização, e assim ser resolvidos com técnicas de regressão e/ou otimização.

Clusterização
Aprendizagem Não Supervisionada



Classificação - Imagens
Aprendizagem Supervisionada

Classificação - Tabelas
Aprendizagem Supervisionada

Tipo de problemas

Essas são as bibliotecas que possivelmente serão utilizadas para resolver o tipo de problema apresentado. Mas note que diversos problemas podem ser consolidados em estruturas de Regressão e/ou Otimização, e assim ser resolvidos com técnicas de regressão e/ou otimização.

Series Temporais

dmlc XGBoost

The Alan Turing Institute

SciPy

Otimização

Particular Design

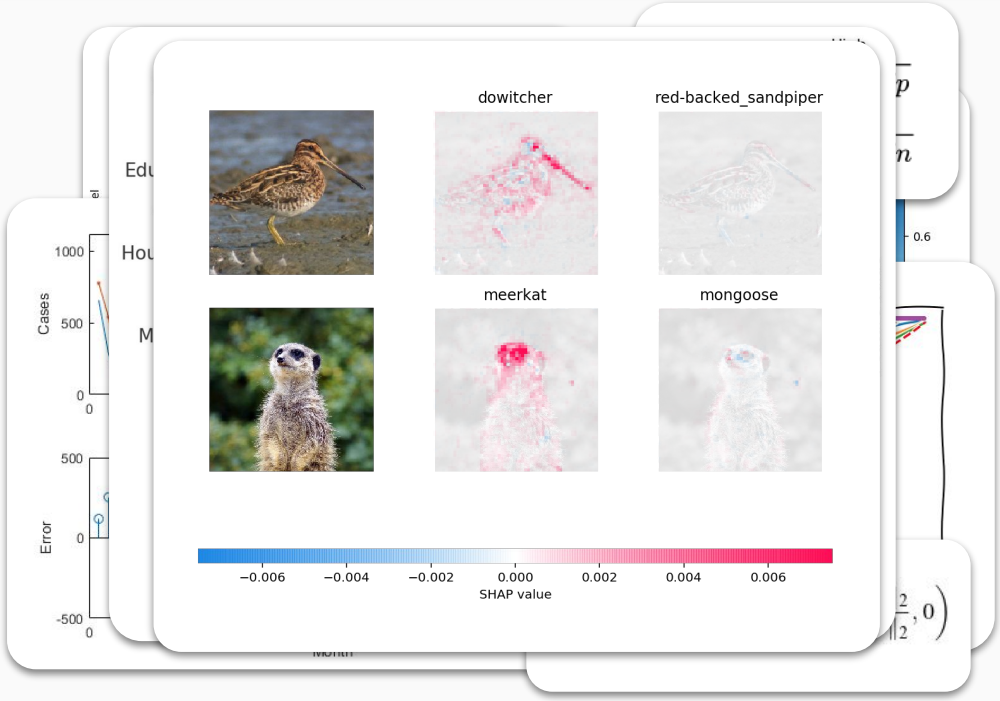
Reinforcement Learning

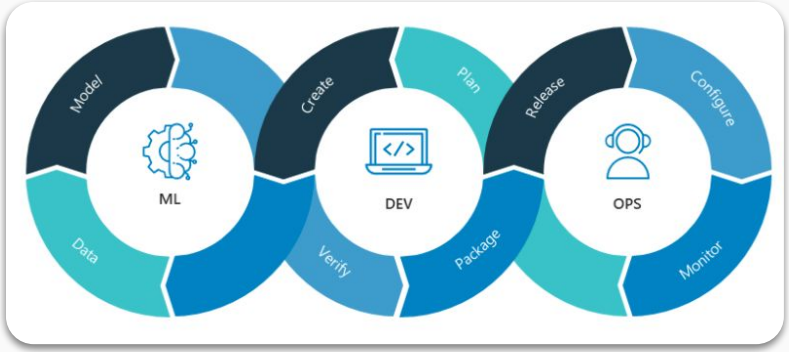
Aprendizagem Supervisionada

Validação

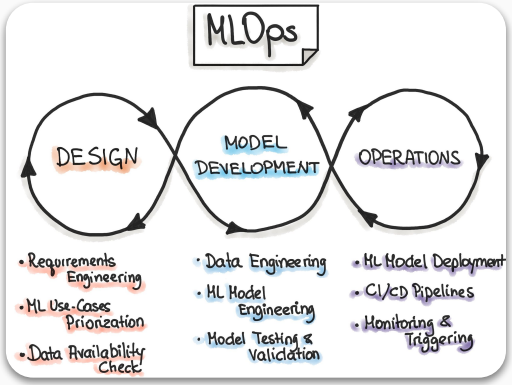
Quais os tipos de validação que normalmente aplicamos em modelos de machine learning?

- Matriz de confusão
- ROC-AUC Curve
- Precisão e Recall
- R2 Score
- RMSE
- Best Fit Rate
- SHAP




Model
Create
Plan
Release
Configure
ML
DEV
OPS
Data
Verify
Package
Monitor


Kedro


MLOps
DESIGN
MODEL DEVELOPMENT
OPERATIONS
• Requirements Engineering
• ML Use-Cases Prioritization
• Data Availability Check
• Data Engineering
• ML Model Engineering
• Model Testing & Validation
• ML Model Deployment
• CI/CD Pipelines
• Monitoring & Triggering


Weights & Biases


SPHINX
Python Documentation Generator

Flask

Note que essa ferramenta serve para você construir seu fornecedor de dados que vai rodar no servidor. Sendo assim sua RestFul API que fica esperando requisições do tipo POST ou GET para realizar alguma tarefa e retornar um resultado para o usuário. Essa tarefa pode ser o que qualquer processo em Python, como por exemplo: acessar o banco de dados, rodar um algoritmo de machine learning, entre outros.


```python
from flask               import Flask, request, jsonify, Response
from flask_restful       import Resource, Api
from flask_request_params import bind_request_params
from flask_cors          import CORS

# Initialize the restful app
app = Flask(__name__)
CORS(app) # Setting App CORS
api = Api(app) # Creating API from App
# Bind request parameters to Application
app.before_request(bind_request_params)

class SumNumbersAPI(Resource):

    def get(self):

        # Build API Response
        response = Response(mimetype='application/json')
        response.headers["Access-Control-Allow-Origin"] = "*"
        response.status_code = 400

        # Make your process
        response.response = sum_numbers(20, 10)
        return response

    def post(self):

        # Build API Response
        response = Response(mimetype='application/json')
        response.headers["Access-Control-Allow-Origin"] = "*"
        response.status_code = 400

        # Get request body
        body_data = json.load(request.data)

        # Make your process
        response.response = sum_numbers(body_data[0], body_data[1])
        return response

# Create each resource...
api.add_resource(SumNumbersAPI, '/sum_numbers')

if __name__ == '__main__':
  app.run(port = 5000, debug = True)
```

Vamos para um exemplo?

Muito obrigado!
Dúvidas?