

# THE DEVELOPER'S CONFERENCE

Machine Learning Efetivo com AWS:  
Da extração dos dados ao deploy em produção

BRUNO BITENCOURT  
Software Architect

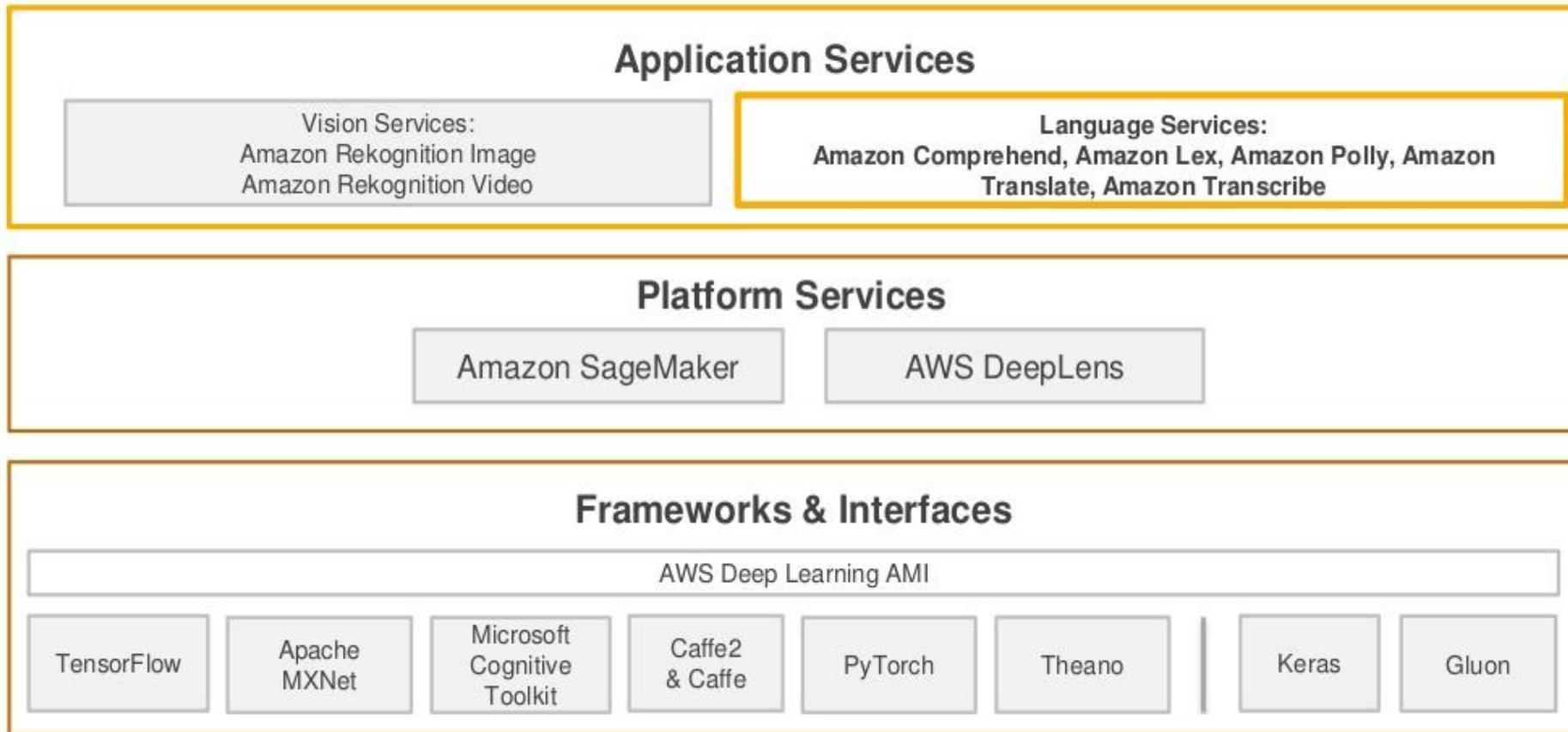
# Agenda

- Machine Learning e AWS
- SageMaker
- Dicas
- Links úteis

# @brunowdev

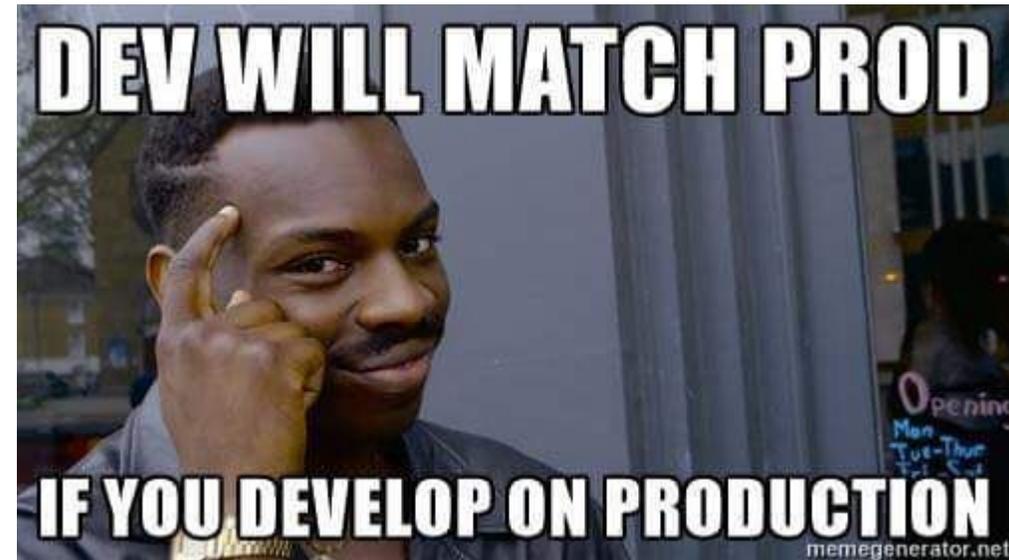
- Arquiteto de Software - P&D da Betha Sistemas
- +5 anos experiencia
- Aluno de Engenharia da Computação na FASATC e Machine Learning Engineer na Udacity
- brunowdev@gmail.com

# Machine Learning e AWS

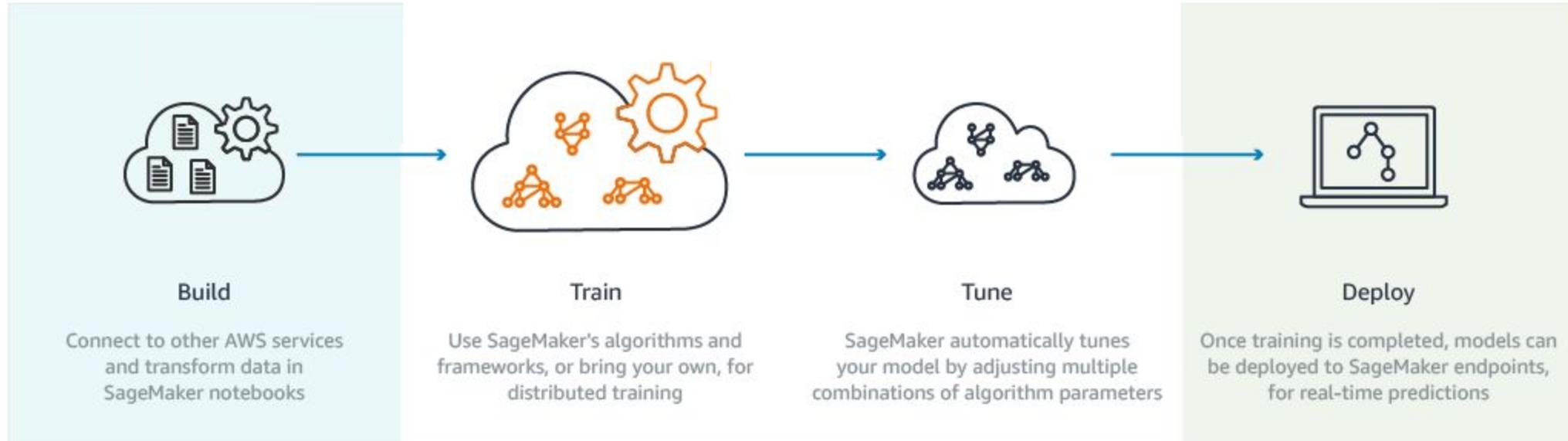




Quanta coisa,  
não é mesmo?



# SageMaker



# Treinamento

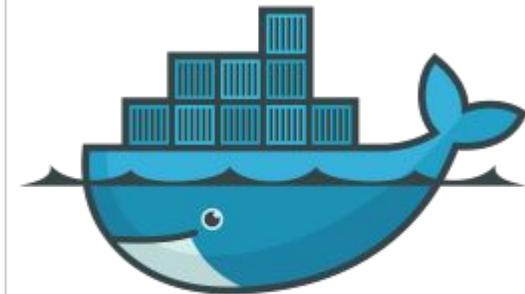
Factorization Machines  
Linear Regression  
PCA  
K-Means  
XGBoost

E vários outros!

Algoritmos Built-in



Traga a sua Impl.



docker

Traga seu container

# Interagindo com o SageMaker

## Python SDK

- Notebooks
  - Algoritmos
  - Treinamento
  - Deploy
  - Tuning
- **AWS CLI:** 'aws sagemaker'
  - **AWS SDK:** boto3, entre outros.
  - Interface WEB

Mas antes disso...

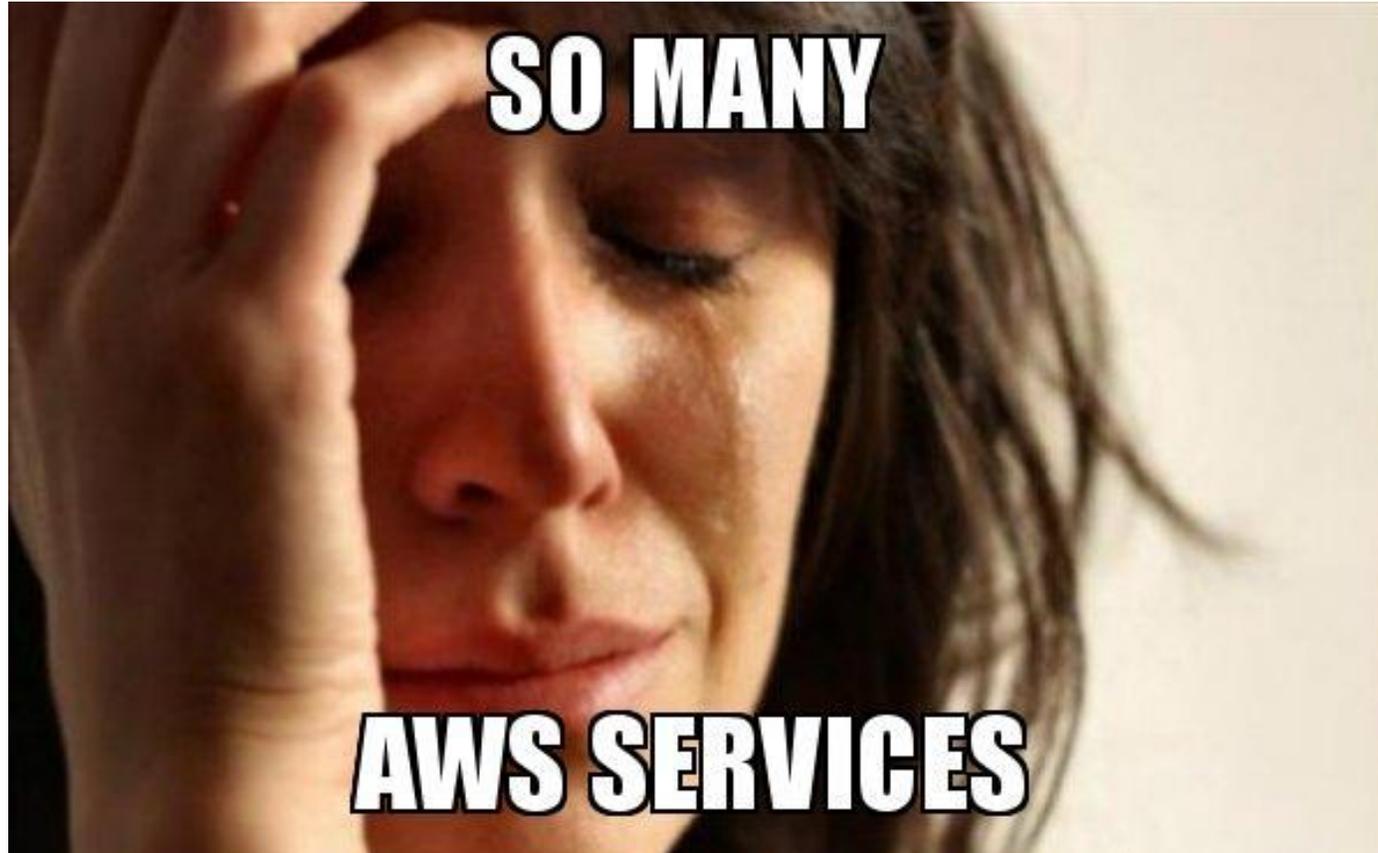


# Etapa 0: Dados

- SQL
- NoSQL
- Aquele dataset do Governo com 500GB
- Outros sistemas
- Data Lake?

# Pipeline de Machine Learning Serverless





# S3

- Buckets para armazenamento
- Big Data
- Velocidade na escrita/acesso
- Também armazenamos nossos modelos de ML



# Glue

- Catalogo de dados (S3, SQL, NoSQL)
- Serverless ETL
- Jobs de extração
  - Python (Spark ou Shell)
  - Scala
- Segurança e parametrização
- Storage no S3



# Job ETL

**Name**  
oracle-rds-etl-production

**IAM role** ⓘ  
rds-orcl-production

Ensure that this role has permission to your Amazon S3 sources, targets, tempor

**Type**  
Spark

**This job runs**

- A proposed script generated by AWS Glue ⓘ
- An existing script that you provide
- A new script to be authored by you

**ETL language**

- Python
- Scala

# Scheduling

**Name**  
run-extraction

▶ **Tags (optional)**

**Trigger type**

Schedule  Job events  On-demand

Choose: **Schedule** to fire the trigger on a timer, **Job events** to fire the trigger when job events match your watched list, **On-demand** to fire the trigger immediately when started.

**Frequency**  
Daily

**Start Hour (UTC)** : **Start Minute**

02 : 00 ⓘ

```
21 env = args['env']
22 url = args['db_url']
23 user = args['db_user']
24 password = args['db_password']
25 output_bucket = args['bucket']
26
27 # template JDBC
28 _data_frame_reader_ = sparkSession.read.format("jdbc").option("url", url) \
29     .option("user", user) \
30     .option("password", password) \
31     .option("driver", "oracle.jdbc.driver.OracleDriver")
32
33 def _table(t_name):
34     return _data_frame_reader_.option("dbtable", t_name)
35
36 funcionarios = _table("funcionarios").load()
37
38 pessoas_fisicas = _table("pessoas_fisicas").load()
39
40 sql_funcionarios = """
41 SELECT f.nome, f.sexo, f.dt_admissao, f.vale_transporte, f.plano_saude, f.estado_civil, pf.dt_nascimento
42 FROM funcionarios f
43 join pessoas_fisicas pf on f.id_pessoa_fisica = pf.id"""
44 funcionarios_df = sparkSession.sql(sql_funcionarios)
```

SQL

```
45
46 funcionarios_df.withColumn('VALE_TRANSPORTE', \
47     when( funcionarios_df['VALE_TRANSPORTE'].isin('S', 'N'), col('VALE_TRANSPORTE')).otherwise('N'))
48
49 funcionarios_df.withColumn('POSSUI_PLANO_SAUDE', \
50     when( funcionarios_df['POSSUI_PLANO_SAUDE'].isin('S', 'N'), col('POSSUI_PLANO_SAUDE')).otherwise('N'))
51
52 funcionarios_df = funcionarios_df.withColumn('SEXO', \
53     when( funcionarios_df['SEXO'].isin('M') == True, 1).otherwise(0)) \
54     .withColumnRenamed('SEXO', 'MASCULINO')
55
56 funcionarios_df.withColumn('ESTADO_CIVIL', \
57     when( isnan(funcionarios_df['ESTADO_CIVIL']), col('ESTADO_CIVIL')).otherwise(1))
58
59 ##### Armazena no S3
60 dailly_folder = 's3://{}/{}'/format(output_bucket, 'extraction', int(time()))
61
62 print('escrevendo arquivos: {}'.formatdailly_folder))
63
64 funcionarios_df.write.format('com.databricks.spark.csv') \
65     .save('{}'/format(dailly_folder, env), header = 'true', delimiter = ';')
66
67 print('orlc-etl: sucesso')
```

Particionamento no S3

Eventos: Sucesso, Falha, etc.

## #TheDevConf 2019

<input type="checkbox"/>	Name ▼
<input type="checkbox"/>	 part-00000-365eba2c-289c-49c2-aada-ffdf75b6c10a-c000.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c000.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c001.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c0098.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c022.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c0481.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c078.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c098.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c123.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c4851.csv
<input type="checkbox"/>	 part-00000-7d8bdbdc-3da4-4d47-8e7f-c3c4aae02834-c954.csv

# Etapa 1: Usando o Dataset



# Infos

- Standard - Sem GPU
- Compute Optimized - Sem GPU
- GPU Instances - Com GPU - inicia \$ 1.26 hora
- O disco varia de 5GB SSD ate 16TB
- Não é obrigatório
- Scripts de Setup e Integração com o Git (não é git friendly)

Notebook instance name

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

Elastic Inference [Learn more](#) 

▼ Additional configuration

Lifecycle configuration - *optional*

Customize your notebook environment with default scripts and plugins.

Volume size in GB - *optional*

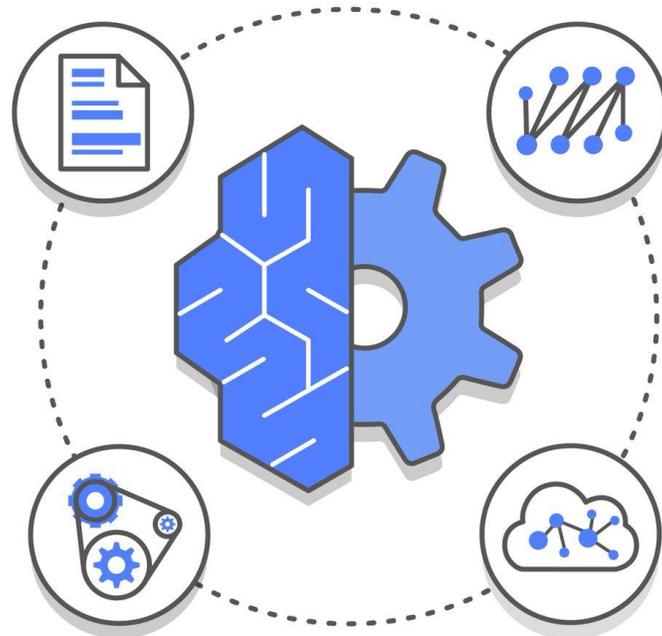
Enter the volume size of the notebook instance in GB. The volume size must be from 5 GB to 16384 GB (16 TB).

GPU

Memory	Pricing
1 GB	\$0.130 per hour
2 GB	\$0.260 per hour
4 GB	\$0.520 per hour

Tensorflow e MxNet

## Etapa 2: SageMaker



# Infos

- Imagens Docker - Train mode e Invocations (HTTP)
- Serverless ou Endpoint
- Qualquer linguagem ou framework
- Lê os dados do S3
- Escreve os modelos no S3

# Devo fazer o meu container?

- Os modelos da AWS são mais otimizados
- Podem ser uma caixa “preta”
- Vendor lock-in

**10X FASTER**  
PERFORMANCE  
THAN ANYWHERE ELSE WITH THE BUILT-IN ALGORITHMS



- Common Elements of Built-in Algorithms
- BlazingText Algorithm
- DeepAR Forecasting Algorithm
- Factorization Machines Algorithm
- Image Classification Algorithm
- IP Insights Algorithm
- K-Means Algorithm
- K-Nearest Neighbors (k-NN) Algorithm
- Latent Dirichlet Allocation (LDA) Algorithm
- Linear Learner Algorithm
- Neural Topic Model (NTM) Algorithm
- Object2Vec Algorithm
- Object Detection Algorithm
- Principal Component Analysis (PCA) Algorithm
- Random Cut Forest (RCF) Algorithm
- Semantic Segmentation Algorithm
- Sequence-to-Sequence Algorithm
- XGBoost Algorithm

# Levando o SciKit Learn para o SageMaker

```
1 FROM ubuntu:16.04
2
3 RUN apt-get -y update && apt-get install -y --no-install-recommends \
4     wget \
5     python \
6     nginx \
7     ca-certificates \
8     && rm -rf /var/lib/apt/lists/*
9
10 RUN wget https://bootstrap.pypa.io/get-pip.py && python get-pip.py && \
11     pip install numpy==1.16.2 scipy==1.2.1 scikit-learn==0.20.2 pandas flask gevent gunicorn && \
12     (cd /usr/local/lib/python2.7/dist-packages/scipy/.libs; rm *; ln ../../numpy/.libs/* .) && \
13     rm -rf /root/.cache
14
15 ENV PYTHONUNBUFFERED=TRUE
16 ENV PYTHONDONTWRITEBYTECODE=TRUE
17 ENV PATH="/opt/program:${PATH}"
18
19 COPY decision_trees /opt/program
20 WORKDIR /opt/program
```

# Exemplo - Treinamento

```
1  #!/usr/bin/env python
2  from __future__ import print_function
3
4  import os
5  import json
6  import pickle
7  import sys
8  import traceback
9
10 import pandas as pd
11
12 from sklearn import tree
13
14 prefix = '/opt/ml/'
15
16 input_path = prefix + 'input/data'
17 output_path = os.path.join(prefix, 'output')
18 model_path = os.path.join(prefix, 'model')
19 param_path = os.path.join(prefix, 'input/config/hyperparameters.json')
20
21 channel_name='training'
22 training_path = os.path.join(input_path, channel_name)
23
```

# Exemplo - Treinamento

```
23
24 def train():
25     print('Iniciando o treinamento.')
26     try:
27
28         with open(param_path, 'r') as tc:
29             trainingParams = json.load(tc)
30
31         input_files = [ os.path.join(training_path, file) for file in os.listdir(training_path) ]
32         if len(input_files) == 0:
33             raise ValueError(('Nao foram encontrados arquivos no diretorio {}. \n' +
34                               'Verifique se o channel ({} ) foi especificado corretamente.').format(training_path, channel_name))
35
36         raw_data = [ pd.read_csv(file, header=None) for file in input_files ]
37         train_data = pd.concat(raw_data)
38
39         train_y = train_data.ix[:,0]
40         train_X = train_data.ix[:,1:]
41
42         # utiliza um hiperparametro, caso tenha sido fornecido
43         max_leaf_nodes = trainingParams.get('max_leaf_nodes', None)
44         if max_leaf_nodes is not None:
45             max_leaf_nodes = int(max_leaf_nodes)
```

## Exemplo - Treinamento

```
# treina o modelo
clf = tree.DecisionTreeClassifier(max_leaf_nodes=max_leaf_nodes)
clf = clf.fit(train_X, train_y)

# salva o modelo no S3
with open(os.path.join(model_path, 'decision-tree-model.pkl'), 'w') as out:
    pickle.dump(clf, out)
print('Treino completo.')

except Exception as e:
    trc = traceback.format_exc()
    with open(os.path.join(output_path, 'failure'), 'w') as s:
        s.write('Erro durante o treinamento: ' + str(e) + '\n' + trc)
    # todos os prints sao adicionados automaticamente nos logs
    print('Erro durante o treinamento: ' + str(e) + '\n' + trc, file=sys.stderr)
    # marca a job como falha
    sys.exit(255)
```

## Exemplo - Predictor

```
@app.route('/ping', methods=['GET'])
def ping():
    """Determina se a instancia esta saudavel."""
    health = ScoringService.get_model() is not None

    status = 200 if health else 404
    return flask.Response(response='\n', status=status, mimetype='application/json')
```

## Exemplo - Predictor

```
class ScoringService(object):
    model = None

    @classmethod
    def get_model(cls):
        """Carrega o modelo uma unica vez."""
        if cls.model == None:
            with open(os.path.join(model_path, 'decision-tree-model.pkl'), 'r') as inp:
                cls.model = pickle.load(inp)
        return cls.model

    @classmethod
    def predict(cls, input):
        """Realiza as predicoes e retorna.

        Args:
            input (pandas dataframe): Os dados para fazer a predicao"""
        clf = cls.get_model()
        return clf.predict(input)
```

## Exemplo - Predictor

```
@app.route('/invocations', methods=['POST'])
def transformation():

    data = None

    if flask.request.content_type == 'text/csv':
        data = flask.request.data.decode('utf-8')
        s = StringIO.StringIO(data)
        data = pd.read_csv(s, header=None)
    else:
        return flask.Response(response='Apenas arquivos no formato CSV sao suportados.', status=415, mimetype='text/plain')

    print('Predicao para {} registros'.format(data.shape[0]))

    data.drop(data.columns[[0]],axis=1,inplace=True)

    predictions = ScoringService.predict(data)

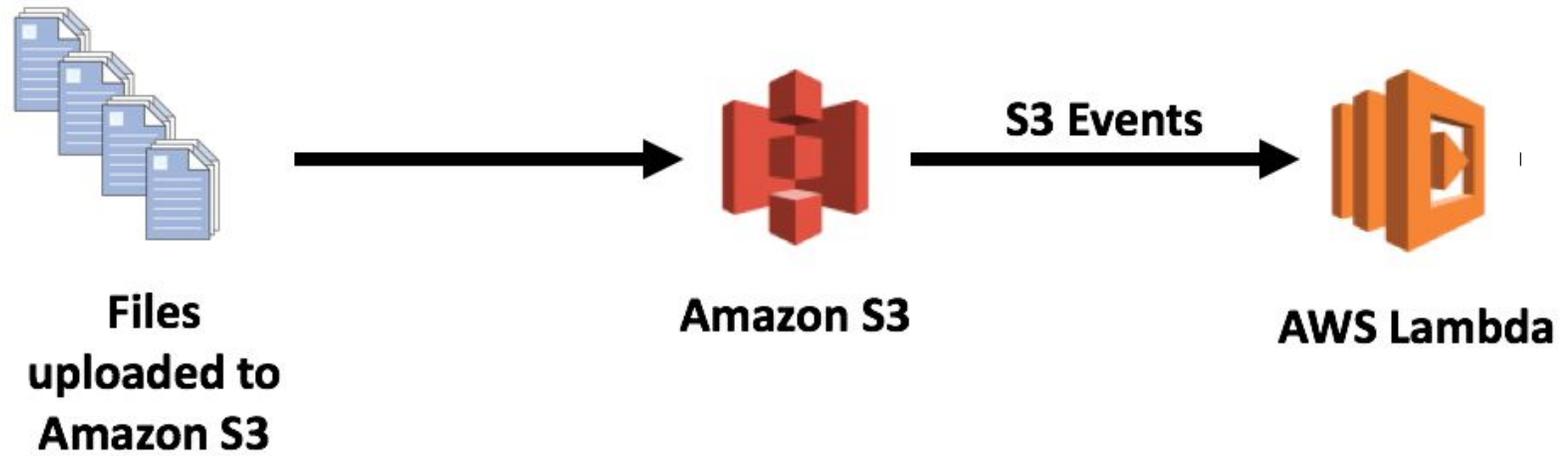
    out = StringIO.StringIO()
    pd.DataFrame({'results':predictions}).to_csv(out, header=False, index=False)
    result = out.getvalue()

    return flask.Response(response=result, status=200, mimetype='text/csv')
```

# Treinando o modelo

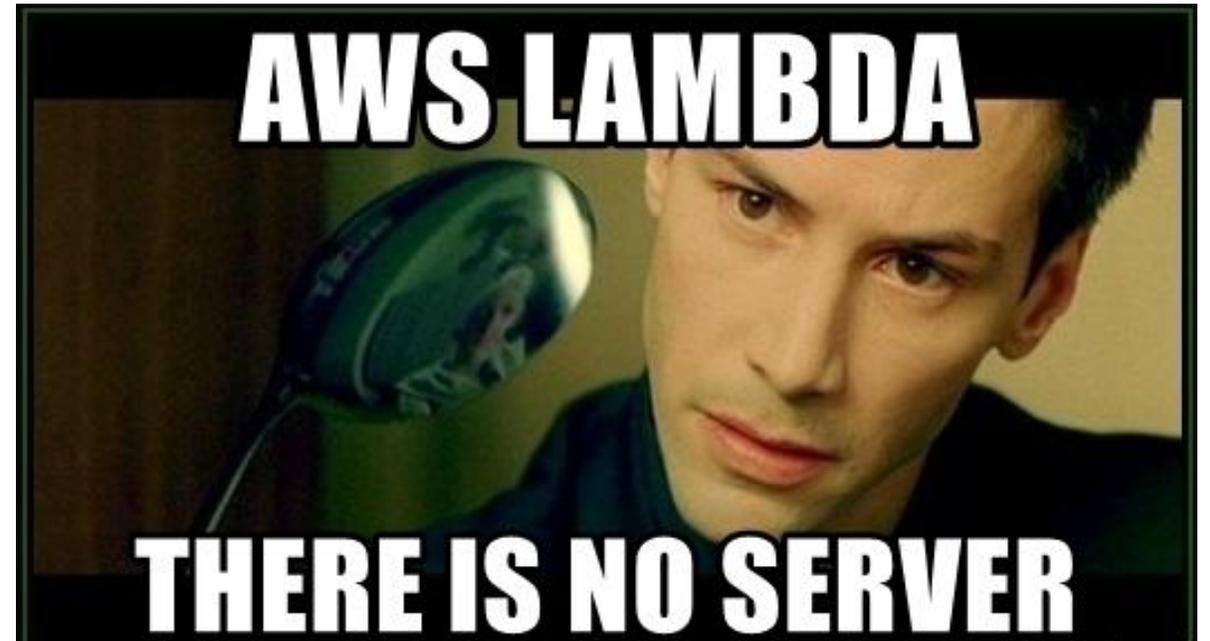
- Manual
- Baseado em eventos

# Treinando o modelo



# AWS Lambda

- Funções Serverless
- Várias linguagens disponíveis
- Integração com eventos
- Baixo custo
- Comunicação direta com os demais serviços





# Rastreabilidade

URI

<s3://supervised-sm/extractions-15482365/>

Model data location

<s3://supervised-sm-poc/output/model.tar.gz> 

Image

[145599724414.dkr.ecr.us-east-1.amazonaws.com/supervised-poc-ex:latest](https://145599724414.dkr.ecr.us-east-1.amazonaws.com/supervised-poc-ex:latest)

▶	02:32:31	Tempo de treinamento: 0.1063
▶	02:32:31	Accuracy treinamento: 0.9886
▶	02:32:31	F-score treinamento: 0.9818
▶	02:32:31	-----
▶	02:32:31	Accuracy teste: 0.9553
▶	02:32:31	F-score teste: 0.9305
▶	02:32:31	Matriz de confusão:
▶	02:32:31	[[4917 413] [ 28 4502]]
▶	02:32:32	Treinamento completo.



Amazon  
CloudWatch

# Hyperparameter Tuning

- Semelhante a job de treinamento
- Qualquer modelo
- Warmup (incremental iterativo ou outra versão)

# Hyperparameter Tuning

```
hyperparameter_ranges = {'degree': IntegerParameter(1, 3),  
                           'thresh': ContinuousParameter(0.001, 0.01),  
                           'prune': CategoricalParameter(['TRUE', 'FALSE'])}  
  
objective_metric_name = 'mse'  
metric_definitions = [{'Name': 'mse',  
                        'Regex': 'mse: ([0-9\\.]+)'}]  
  
tuner = HyperparameterTuner(estimator,  
                             objective_metric_name,  
                             hyperparameter_ranges,  
                             metric_definitions,  
                             objective_type='Minimize',  
                             max_jobs=9,  
                             max_parallel_jobs=3)
```

# Deploy

- Endpoint Configuration - Middleman
  - Roteia com base em headers (id de usuário, região, etc.)
- Sem Downtime
- Auto Scaling
- Canary Deployment
- Teste A/B
  - Monitoramento/Scoring é manual

# Inferência

- Endpoint SageMaker - Requer uma instância rodando
- Batch Transform Jobs

```
response = sagemaker.create_transform_job(  
    TransformJobName = 'batch-tranform-{}-{}'.format(database, int(time.time())),  
    ModelName = latest_model['ModelName'],  
    MaxConcurrentTransforms = 40,  
    MaxPayloadInMB = 6,  
    BatchStrategy = 'MultiRecord',  
    Environment = {  
        'ENV': ENV  
    },  
    TransformInput = {  
        'DataSource': {  
            'S3DataSource': {  
                'S3DataType': 'S3Prefix',  
                'S3Uri': 's3://supervised-sm/extractions/to-predict/'.format(key),  
            }  
        },  
        'ContentType': 'application/json',  
        'CompressionType': 'None',  
        'SplitType': 'None'  
    },  
    TransformOutput = {  
        'S3OutputPath': 's3://supervised-sm/extractions/output{}'.format(extraction_key),  
        'Accept': 'application/json'  
    },  
    TransformResources={  
        'InstanceType': 'ml.m5.large',  
        'InstanceCount': 3  
    },  
    Tags=[  
        {  
            'Key': 'ENV',  
            'Value': ENV  
        },  
    ],  
)
```

# Consumindo as Predições

```
response = s3.get_object(Bucket = 'supervised-sm', Key = key)
result = json.loads(response['Body'].read().decode('utf-8'))

predictions = result['predictions']

responses = []
for prediction in predictions:

    response = sqs.send_message(
        QueueUrl = queue_url,
        MessageBody = json.dumps(prediction),
        DelaySeconds = 0,
        MessageAttributes = {
            'string': {
                'StringValue': 'recomendado-para-voce',
                'DataType': 'String'
            }
        },
        MessageGroupId = 'recommendation-result'
    )
```

# Considerações

- Para gerenciar os componentes: Terraform, Cloudformation:
  - Todas se integram com facilidade com CI (Jenkins, Gitlab, etc.)

# Dicas

- Use só as peças que você precisa
- Desligue os seus notebooks (pode ser automatizado)
- Ajuste as instâncias de treino/predição de acordo com a sua necessidade
- Use as imagens/algoritmos disponíveis, sempre que possível
- Use RecordIO / TFRecord

# Outros recursos

## **GroundTruth**

- Automatizar o processo de labelling

## **Marketplace**

- Publique e venda seus modelos

# Outras informações

## SageMaker

- Possui free-tier
- Ainda não está disponível na **sa-east-1**
  - Latência pode ser um problema

## Amazon SageMaker

**250 hours** per month of t2.medium notebook usage for the first two months

**50 hours** per month of m4.xlarge for training for the first two months

**125 hours** per month of m4.xlarge for hosting for the first two months

# Por onde começar?

## AWS

- [Getting Started](#)
- [Playlist - This is my Architecture](#)

## SageMaker

- [Julien Simon](#)
- [Exemplos GitHub](#)

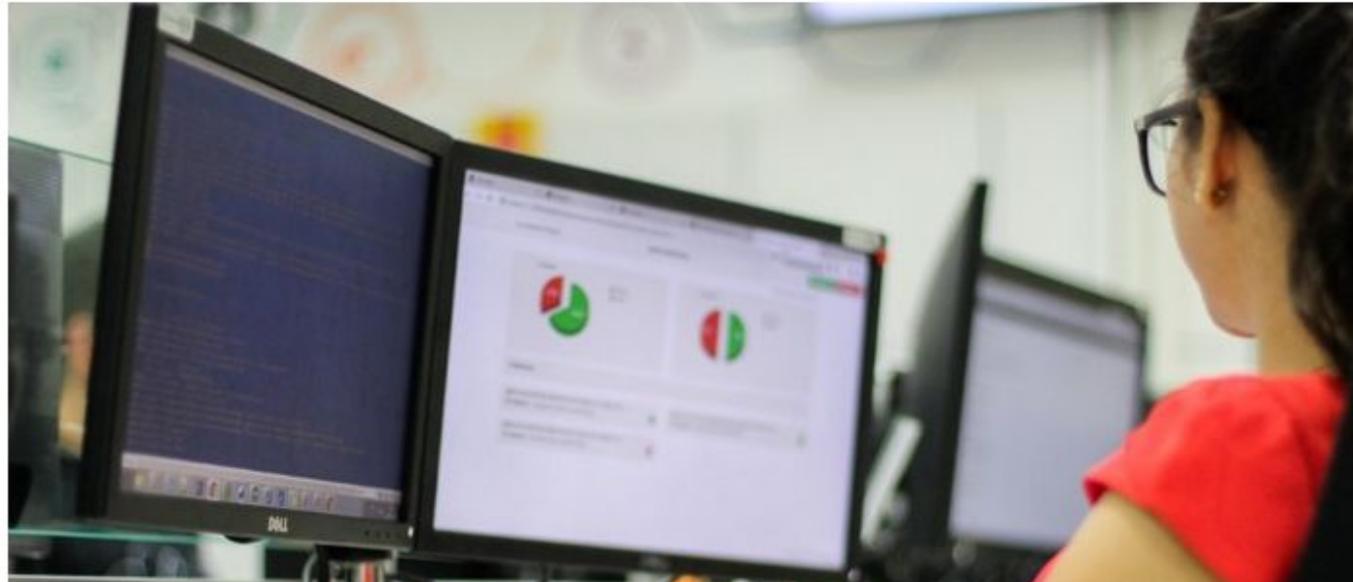


# Nosso Case

## EDUCAÇÃO

Com Machine Learning, Betha Sistemas prevê reprovação e evasão escolar logo no início do ano

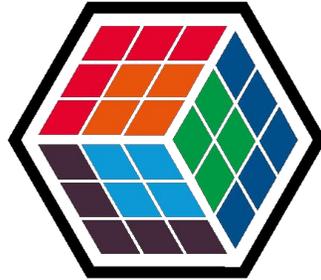
Utilizando Inteligência Artificial, nove em cada 10 evasões poderiam ser evitadas



Perguntas?



*“That’s all Folks!”*



# THE DEVELOPER'S CONFERENCE