

PROJET 8



Fruits!

PRÉSENTÉ PAR LOÏC
VALENTI

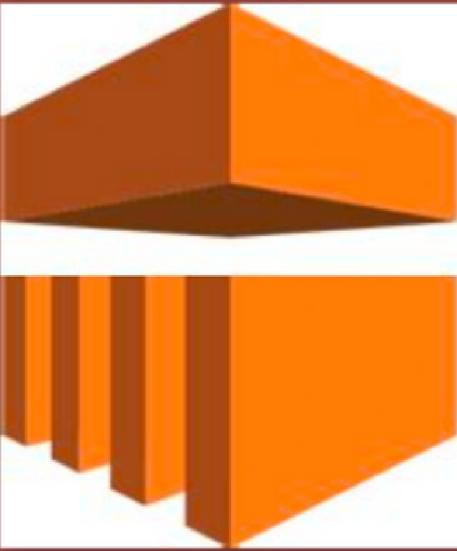


FRUITS

APERÇU DU RAPPORT

Implémentation de pré-traitement de classification d'image sur le cloud.

FRUIT



amazon
EMR



Fruits 360

A dataset with 90380 images of 131 fruits and vegetables



03

Fruits 360

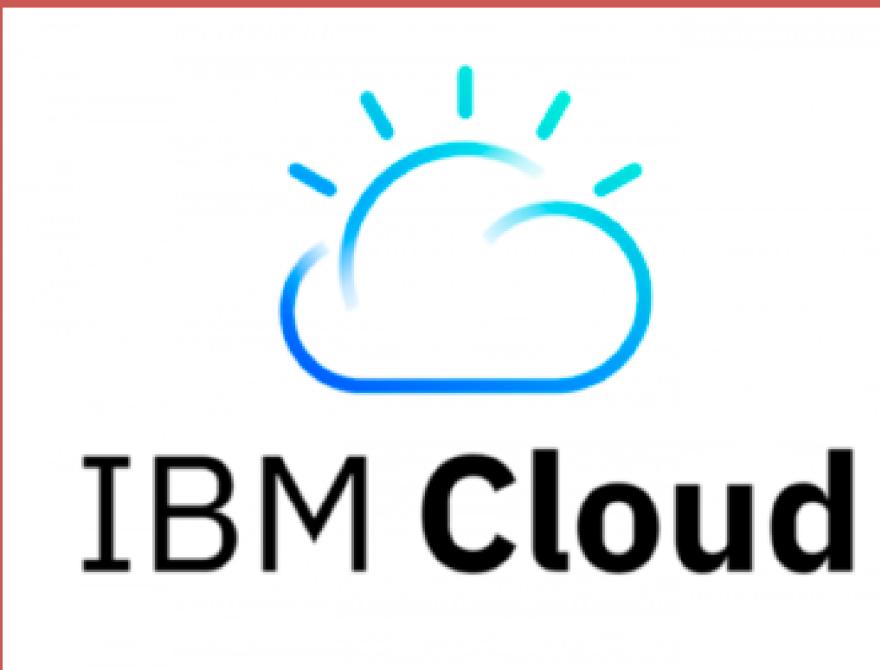
A dataset with 90380 images of 131 fruits and vegetables



05

LE CLOUD ?

Des serveurs informatiques à distance et hébergés sur internet pour stocker, gérer et traiter des données, plutôt qu'un serveur local ou un ordinateur personnel



AWS ? AMAZON WEB SERVICES



- Services de cloud computing à la demande
- AWS propose plus de 90 services, comprenant le calcul, le stockage, le réseau

– WIKIPEDIA

07

CRÉATION DE L'ENVIRONNEMENT BIG DATA

1. Inscription sur AWS
2. Enregistrer les données sur S3
3. Créer l'environnement EMR
4. Exécuter les scripts PySpark
5. Enregistrer les données pré-traitées

INSCRIPTION SUR AWS



S'inscrire à AWS

- Inscription gratuite
- Requiert un numéro de téléphone
- Requiert une carte bancaire valide

Adresse e-mail de l'utilisateur racine
Utilisé pour la récupération de compte et certaines fonctions administratives

Nom du compte AWS
Choisissez le nom de votre compte. Vous pouvez modifier ce nom dans les paramètres de votre compte après l'inscription.

08

Vérifier l'adresse e-mail

ENREGISTRER LES DONNÉES SUR S3



- Storage permanent
- Accessible depuis une multitude de plateforme (CLI, web UI)
- 0,023 USD par Go par mois, H.T

Amazon S3

Stockez et récupérez n'importe quelle quantité de données, n'importe où

Amazon S3 est un service de stockage d'objet offrant une capacité de mise à l'échelle, une disponibilité des données, une sécurité et des performances de pointe.

ENREGISTRER LES DONNÉES SUR S3



Stockage du jeu de données

dans le bucket p8-2023

Compartiments (2) Info		
Les compartiments sont des conteneurs pour les données stockées dans S3. En savoir plus		
	Nom	Région AWS
<input type="radio"/>	aws-logs-002405489643-eu-west-3	EU (Paris) eu-west-3
<input type="radio"/>	p8-2023	EU (Paris) eu-west-3

CRÉER L'ENVIRONNEMENT EMR



- Serveurs customisables
- Toujours disponibles
- Capacités de calcul réglable
- Directement relié à S3
- Logiciels pré-installés

Amazon EMR

Exécutez et mettez à l'échelle facilement Apache Spark, Apache Hive, Presto et d'autres charges de travail big data.

CRÉER L'ENVIRONNEMENT EMR

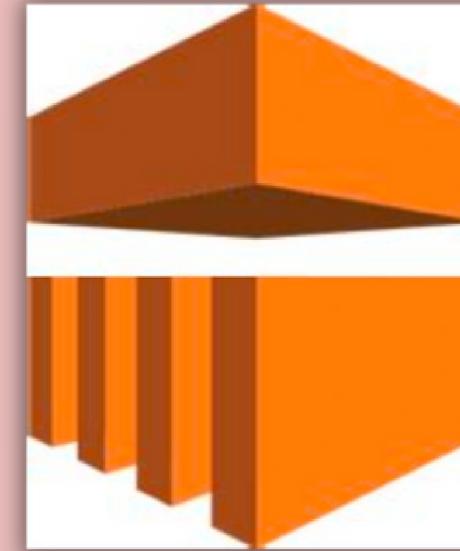


amazon
EMR

- Un noeud maître m5.xlarge X
- Deux noeuds workers 4 vCore, 16 Gio de mémoire, Stockage EBS uniquement
- 0,192 USD par heure par noeud Stockage EBS
m5.xlarge, H.T 64 GiB

13

CRÉER L'ENVIRONNEMENT EMR



amazon
EMR

Applications

- ~20 minutes d'initialisation
- Facturé dès le lancement
- Possibilité de bootstrap les noeuds avec des logiciels

Version d'Amazon EMR
emr-6.3.0

Installed applications
Hadoop 3.2.1, Hive 3.1.2, Pig 0.17.0, Hue
4.9.0, JupyterHub 1.2.0, Spark 3.1.1,
TensorFlow 2.4.1

14



RECAP



- Inscrit sur AWS
- Enregistrement des données dans le bucket p8-2023
- Instanciation du cluster avec les logiciels requis installés

CONNECTION INTERFACE WEB SSH



What is Apache Spark?

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size. It provides development APIs in Java, Scala, Python and R, and supports code reuse across multiple workloads—batch processing, interactive queries, real-time analytics, machine learning, and graph processing.

PYSPARK

PYSPARK

- Spark in python



EXÉCUTER LES SCRIPTS PYSPARK.

INITIALISATION DE SPARKCONTEXT

18

PYSPARK

- Spark in python



```
Entrée [*]: import pandas as pd
import numpy as np
import io
import os
import tensorflow as tf
from PIL import Image
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras import Model
from pyspark.sql.functions import col, pandas_udf, PandasUDFType, element_at, split
```

Starting Spark application

4.10.4 Définition des PATH pour charger les images et enregistrer les résultats

Nous accédons directement à nos données sur S3 comme si elles étaient stockées localement.

```
Entrée [ ]: PATH = 's3://p8-2023'
PATH_Data = PATH+'/Test'
PATH_Result = PATH+'Results_2/Results_final'
print('PATH:           '+\
      PATH+'\nPATH_Data:   '+\
      PATH_Data+'\nPATH_Result: '+PATH_Result)
```

EXÉCUTER LES SCRIPTS PYSPARK.

LECTURE DES IMAGES

19

4.10.5.1 Chargement des données

```
Entrée [*]: images = spark.read.format("binaryFile") \
    .option("pathGlobFilter", "*.jpg") \
    .option("recursiveFileLookup", "true") \
    .load(PATH_Data)
```

```
Entrée [ ]: images.show(5)
```

Je ne conserve que le **path** de l'image et j'ajoute
une colonne contenant les **labels** de chaque image :

```
Entrée [ ]: images = images.withColumn('label', element_at(split(images['path'], '/'), -2))
print(images.printSchema())
print(images.select('path', 'label').show(5, False))
```

Modèle pré-entraîné par google pour haute performance sur mobiles.

Nombre de paramètres réduits, performance optimisé.

Total params: 2,257,984

Trainable params: 2,223,872

Non-trainable params: 34,112

EXÉCUTER LES SCRIPTS PYSPARK. PRE-PROCESSING DES IMAGES.

Layer (type)	Output Shape	Param #	Connected to
=====			
input_1 (InputLayer)	[(None, 224, 224, 3) 0		

```
def preprocess(content):
    """
    Preprocesses raw image bytes for prediction.
    """
    img = Image.open(io.BytesIO(content)).resize([224, 224])
    arr = img_to_array(img)
    return preprocess_input(arr)
```

EXÉCUTER LES SCRIPTS PYSPARK. INITIALISATION DE MOBILENETV2 ET BROADCASTING DES POIDS.

22

```
def model_fn():
    """
    Returns a MobileNetV2 model with top layer removed
    and broadcasted pretrained weights.
    """
    model = MobileNetV2(weights='imagenet',
                         include_top=True,
                         input_shape=(224, 224, 3))
    for layer in model.layers:
        layer.trainable = False
    new_model = Model(inputs=model.input,
                      outputs=model.layers[-2].output)
    new_model.set_weights(broadcast_weights.value) #Broadcasting the weights
    return new_model
```

EXÉCUTER LES SCRIPTS PYSPARK. FEATURISATION DES DONNÉES.

23

```
Entrée [19]: features_df = images.repartition(24).select(col("path"),
                                                     col("label"),
                                                     featurize_udf("content").alias("features")
                                                    )

Entrée [20]: print(PATH_Result)

s3://p8-2023Results_2/Results_final
```

EXÉCUTER LES SCRIPTS PYSPARK, STD_SCALING

24

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks
driver	ip-172-31-16-30.eu-west-3.compute.internal:42451	Active	0	8.7 MiB / 353.4 MiB	0.0 B	0	0
5	ip-172-31-26-81.eu-west-3.compute.internal:42969	Active	0	42.2 KiB / 4.8 GiB	0.0 B	2	2
6	ip-172-31-20-23.eu-west-3.compute.internal:35921	Active	0	42.2 KiB / 4.8 GiB	0.0 B	2	2

8.5 MINUTES

```
df = features_df.select(  
    array_to_vector('features').alias('features'))  
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures",  
    withStd=True, withMean=True)  
  
# Compute summary statistics by fitting the StandardScaler  
scalerModel = scaler.fit(df)  
  
# Normalize each feature to have unit standard deviation.  
scaledData = scalerModel.transform(df)
```

EXÉCUTER LES SCRIPTS PYSPARK, PCA.

25

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks
driver	ip-172-31-16-30.eu-west-3.compute.internal:42451	Active	0	8.8 MiB / 353.4 MiB	0.0 B	0	0
7	ip-172-31-26-81.eu-west-3.compute.internal:44541	Active	0	42.2 KiB / 4.8 GiB	0.0 B	2	3
8	ip-172-31-20-23.eu-west-3.compute.internal:41815	Active	0	42.2 KiB / 4.8 GiB	0.0 B	2	3

```
pca = PCA(k=200, inputCol="scaledFeatures", outputCol="pcaFeatures")
model = pca.fit(scaledData)

result = model.transform(scaledData)
```

12 MINUTES



ENREGISTRER LES DONNÉES SUR S3

<input type="checkbox"/>	_SUCCESS	-	27 Feb 2023 01:16:35 PM CET	0 o	Standard
<input type="checkbox"/>	part-00000-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:14:14 PM CET	15.6 Mo	Standard
<input type="checkbox"/>	part-00001-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:14:16 PM CET	15.7 Mo	Standard
<input type="checkbox"/>	part-00002-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:14:14 PM CET	15.6 Mo	Standard
<input type="checkbox"/>	part-00003-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:14:16 PM CET	15.5 Mo	Standard
<input type="checkbox"/>	part-00004-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:14:40 PM CET	15.5 Mo	Standard
<input type="checkbox"/>	part-00005-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:14:41 PM CET	15.5 Mo	Standard
<input type="checkbox"/>	part-00006-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:14:44 PM CET	15.6 Mo	Standard
<input type="checkbox"/>	part-00007-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:14:44 PM CET	15.5 Mo	Standard
<input type="checkbox"/>	part-00008-d7f1bfcb-af6b-4139-b50b-2e813984c243-c000.snappy.parquet	parquet	27 Feb 2023 01:15:04 PM CET	15.5 Mo	Standard

9 MINUTES

DEMONSTRATION