

Projet P8

Déployer un modèle dans le
cloud

Introduction

2

- DataScientist pour Fruits! -> jeune start-up de l'AgriTech cherchant à proposer des solutions innovantes pour la récolte des fruits en développant des robot cueilleur intelligent



- **Objectifs :**
 - Développer un environnement Big Data
 - Etablir une première chaine de traitement avec une étape de preprocessing & de réduction des données

Sommaire

3

1. Données
2. Big Data
3. Environnement AWS
4. Script Python/notebook
5. Conclusion

Données

4

Le dataset "Fruit 360" contient un total de 90 483 images

- Training : 67 692 images
- Test : 22 688 images

Repartie en 121 répertoire :

- 1 répertoire = 1 fruit ou légumes
- Plusieurs variétés
- Rotation du fruit sur 3 axes, à 360°
- Taille : 100*100 px

**On va utiliser un échantillon de 21 fruits contenant 6231 images
(fruits-360-original-size/Training)**



5

Intérêt : Lorsque la quantité de donnée ne peut pas être gérée par la RAM de la machine = Problème de BIG DATA

Comment gérer ce problème : En parallélisant les calculs sur différentes machines



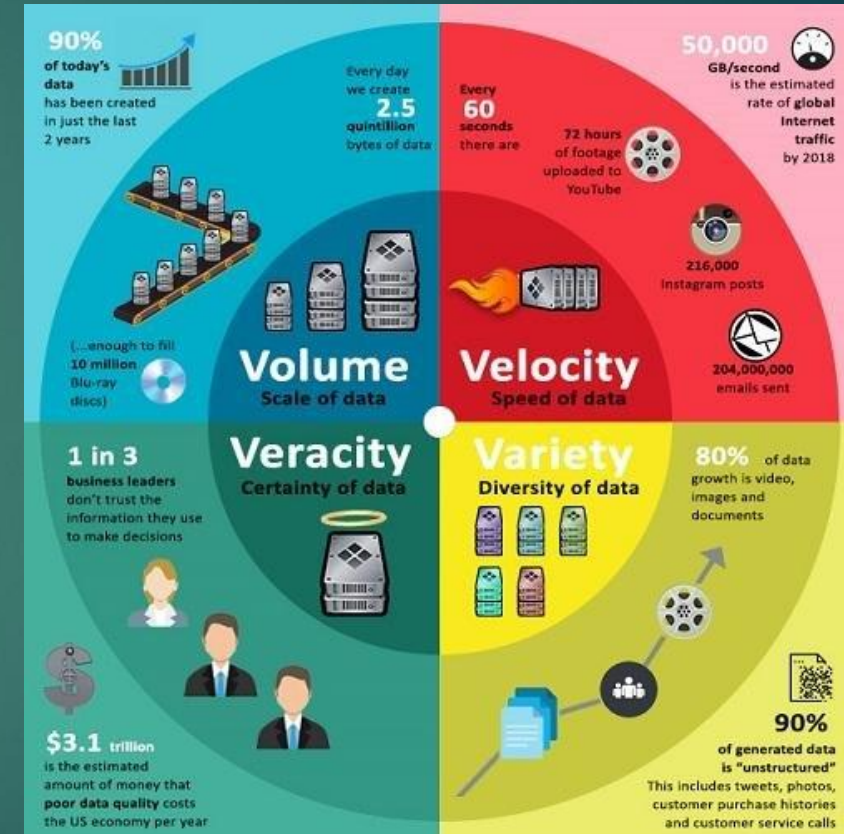
Big Data

6

Enjeux des 4V :

- ▶ Volume : Stockage des données
- ▶ Variété : Structure des données
- ▶ Vitesse : Traitement en temps réel
- ▶ Véracité : Fiabilité et qualité des données

Comment distribuer les calcul ?



Distribution des calculs :

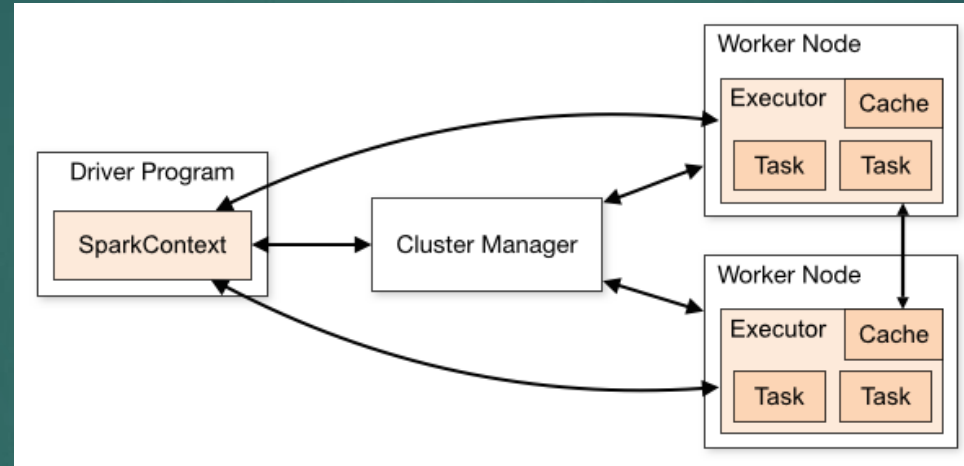
Spark :

- ▶ framework de calcul distribué open source
- ▶ Traitement de larges volumes de données de manière distribuée
- ▶ 100 x plus rapide que Hadoop Map Reduce

PySpark :

- ▶ interface pour Apache Spark sur Python
- ▶ Une alternative plus puissante que Pandas pour le Big Data
- ▶ Prend en charge la plupart des fonctionnalités de Spark

Distribution des calculs :



- **Driver** : Répartie les tâches sur les différents executors
- **Cluster manager** : instancie les différents workers
- **Workers** : instancie un executor chargé d'exécuter les différentes tâches de calculs

Déployer sur le cloud

9

Service AWS pour le Machine Learning :

Workflow Services



Amazon SageMaker



Deep Learning AMI



Deep Learning Containers



AWS Batch



AWS ParallelCluster



Elastic Kubernetes Service



Elastic Container Service



Amazon EMR

Frameworks



TensorFlow

PyTorch

mxnet

Keras



Compute, Networking, and Storage



EC2 Tn1 instances



EC2 P4 instances



EC2 Inf1 instances



EC2 G5 instances



Elastic Inference



AWS Outposts



Elastic Fabric Adapter



Amazon S3



Amazon EBS





Amazon FSx





Amazon EFS

Service AWS pour déployer un modèle :

	 Amazon EC2	 Amazon SageMaker
Avantage	<ul style="list-style-type: none">- Faible coût	<ul style="list-style-type: none">- Adapté au ML- Notebook Jupyter- Simplicité d'utilisation
Inconvénient		<ul style="list-style-type: none">- Coût (version gratuite possible)

Service AWS pour déployer un modèle :

	 Amazon EC2	 Amazon SageMaker
Avantage	- Faible cout	<ul style="list-style-type: none">- Adapté au ML- Notebook Jupyter- Simplicité d'utilisation
Inconvénient		<ul style="list-style-type: none">- Cout (version gratuite possible)

Méthode choisi pour la suite du projet

Sagemaker :

- ▶ SageMaker couvre toutes les étapes de la machine learning: de la collecte jusqu'au déploiement du modèle
- ▶ Création d'un notebook en utilisant une instance t3.medium et relié à notre bucket

Amazon SageMaker > Instances de blocs-notes > p8-opc-aginth

p8-opc-aginth

[Supprimer](#) [Démarrer](#) [Ouvrir Jupyter](#) [Ouvrir JupyterLab](#)

Paramètres d'instances de blocs-notes [Modifier](#)

Nom	Statut	Type d'instance de bloc-notes	Identifiant de plateforme
p8-opc-aginth	⊖ Stopped	ml.t3.medium	Amazon Linux 2, Jupyter Lab 1 (notebook-al2-v1)
ARN	Heure de création	Taille du volume	Version IMDS minimale
arn:aws:sagemaker:eu-west-3:202048666577:notebook-instance/p8-opc-aginth	Dec 07, 2022 16:21 UTC	5GB EBS	1
Configuration du cycle de vie	Date de la dernière mise à jour		
-	Dec 14, 2022 16:34 UTC		






AWS S3 : Simple Storage Service

- ▶ Stockage des données
- ▶ Les buckets = des dossiers (nom doit être unique)
- ▶ Accès avec la librairie boto3

Amazon S3 > Compartiments

► **Instantané de compte** Afficher le tableau de bord de Storage Lens
Storage Lens offre une visibilité sur l'utilisation du stockage et les tendances d'activité. [En savoir plus](#)

Compartiments (3) [Info](#)
Les compartiments sont des conteneurs pour les données stockées dans S3. [En savoir plus](#)

  Copier l'ARN  Vider  Supprimer  Créer un compartiment

	Nom ▲	Région AWS ▼	Accéder ▼	Date de création ▼
<input type="radio"/>	p8-opc-train-aginth	EU (Paris) eu-west-3	Compartiment et objets non publics	01 Dec 2022 06:53:16 PM CET
<input type="radio"/>	p8-resultats	EU (Paris) eu-west-3	Compartiment et objets non publics	11 Dec 2022 12:46:29 PM CET

AWS IAM : Identity and Access Management

- ▶ Permet de gérer les services AWS accessibles à un compte IAM
- ▶ Créé des rôles à partir de stratégies d'autorisations

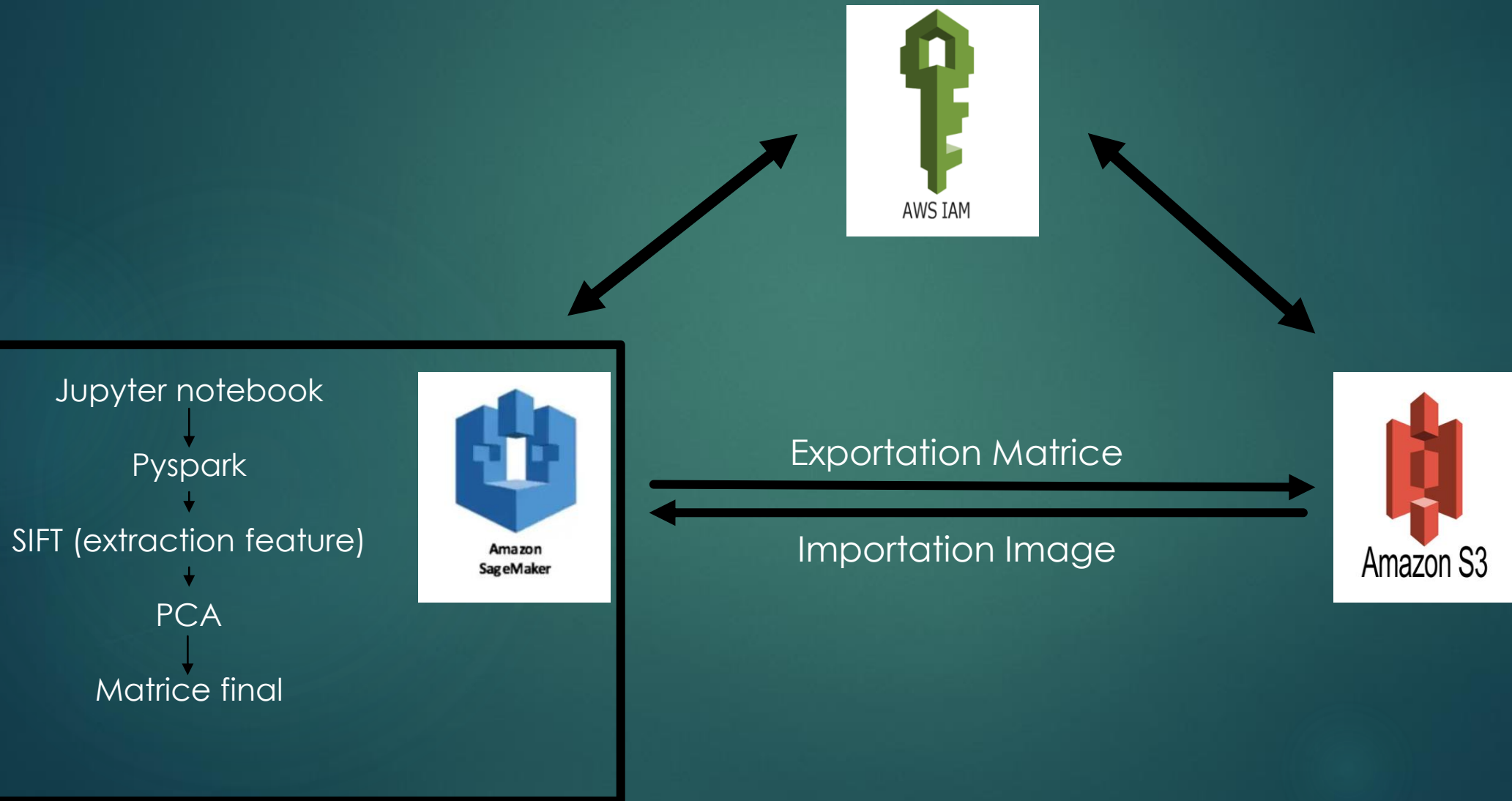
The screenshot displays the AWS IAM console interface. On the left is a navigation sidebar with categories like 'Tableau de bord', 'Gestion des accès', and 'Rapports d'accès'. The main content area shows the 'Récapitulatif' page for the user 'aginth_opc'. At the top, there's a notification about a new feature to generate policies based on CloudTrail events. Below this, a summary section lists the user's ARN, path, and creation time. A tabbed interface allows switching between 'Autorisations', 'Groupes', 'Balises', 'Informations d'identification de sécurité', and 'Access Advisor'. The 'Autorisations' tab is active, showing a section for 'Permissions policies (3 stratégies appliquées)'. This section includes a table of attached policies:

Nom de la stratégie	Type de stratégie	
Attachée directement		
IAMUserChangePassword	Stratégie gérée par AWS	✕
AmazonSageMakerFullAccess	Stratégie gérée par AWS	✕
AmazonS3FullAccess	Stratégie gérée par AWS	✕

Environnement AWS

15

Récapitulatif de l'environnement



Etape 1/

Initialisation de la session Spark

```
session = boto3.session.get_session()
credentials = session.get_credentials()

conf = (SparkConf()
        .set("spark.driver.extraClassPath", ":%s".join(sagemaker_pyspark.classpath_jars())))

spark = (
    SparkSession
    .builder
    .config(conf=conf) \
    .config('fs.s3a.access.key', credentials.access_key) \
    .config('fs.s3a.secret.key', credentials.secret_key) \
    .config("spark.driver.memory", "15g") \
    .master('local[*]') \
    .appName("P8_opc") \
    .getOrCreate()
)

sc = spark.sparkContext
```

executed in 2.06s, finished 19:35:57 2022-12-16

Etape 2/

Import des données depuis notre bucket S3

```
bucket_name = 'p8-opc-train-agingh'
```

```
# Récupération des ressources sur le service AWS S3
s3_client = boto3.client("s3")
s3 = boto3.resource('s3')
bucket = s3.Bucket(bucket_name)
```

Script Python

17

Etape 3/
Extraction features
avec SIFT

```
Entrée [ ]: # METHODE 1 AVEC SIFT

ls_features=dict()
for my_bucket_object in bucket.objects.all():
    if my_bucket_object.key.endswith('.jpg'):
        file_byte_string = s3_client.get_object(Bucket=bucket_name, Key=my_bucket_object.key)['Body'].read()

        np_1d_array = np.frombuffer(file_byte_string, dtype="uint8")
        new_size = (20, 20)
        img = cv2.imdecode(np_1d_array, cv2.IMREAD_COLOR)
        resized = cv2.resize(img, new_size)
        sift = cv2.SIFT_create()
        keypoints, descriptors = sift.detectAndCompute(resized, None)
        if str(type(descriptors)) != "<class 'NoneType'>":
            ls_features[my_bucket_object.key] = descriptors.tolist()[0]
```

Création de la Dataframe PySpark

```
def features_pyspark_df(features_val, features_keys):
    features_df = spark.createDataFrame([(1,) for l in features_val], ['Features'])
    img_df = spark.createDataFrame([(1,) for l in features_keys], ['img_path'])

    img_df = img_df.withColumn("row_idx", row_number().over(Window.orderBy(monotonically_increasing_id()))
    features_df = features_df.withColumn("row_idx", row_number().over(Window.orderBy(monotonically_increasing_id()))

    df_img_feat = img_df.join(features_df, img_df.row_idx == features_df.row_idx).drop("row_idx")
    return df_img_feat
```

```
df_feat = features_pyspark_df(list(ls_features.values()), list(ls_features.keys()))
```

```
df_feat.show(2)
```

```
+-----+-----+-----+
|img_path|Features|row_idx|
+-----+-----+-----+
|Training/apple_6/...|[1.0, 2.0, 3.0, 4...|1|
|Training/apple_6/...|[4.0, 4.0, 1.0, 3...|2|
+-----+-----+-----+
only showing top 2 rows
```

Etape 4/

Script Python

18

Etape 5/

PCA & Reduction de données

```
Entrée [14]: def display_pca_eboulis(pca):
    varexpl = pca.explainedVariance*100

    # Affichage de la variance cumulée
    plt.figure(figsize=(11,7))
    plt.bar(np.arange(len(varexpl))+1, varexpl)

    cumSumVar = varexpl.cumsum()
    plt.plot(np.arange(len(varexpl))+1, cumSumVar,c="red",marker='o')
    plt.axhline(y=90, linestyle="--", color="green",linewidth=1)

    limit = 90
    valid_idx = np.where(cumSumVar >= limit)[0]
    min_plans = valid_idx[cumSumVar[valid_idx].argmin()]
    print('90% de la variance expliqué par', min_plans, "composantes")
    plt.axvline(x=min_plans, linestyle="--",color="green",linewidth=1)

    plt.xlabel("Nb de composante")
    plt.ylabel("Pourcentage d'inertie")
    plt.title("Eboulis des valeurs propres")
    plt.show(block=False)

    return min_plans

Entrée [15]: def PCA (df_feat,nb_composante):
    vector_dense = udf(lambda x: Vectors.dense(x), VectorUDT())
    img_vd_df = df_feat.select('Features', vector_dense("Features").alias("features_vd"))

    # PCA
    pca_spark = pyspark.ml.feature.PCA(inputCol="features_vd", outputCol="features_pca", k=nb_composante)
    pca = pca_spark.fit(img_vd_df)
    min_plans = display_pca_eboulis(pca)

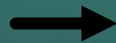
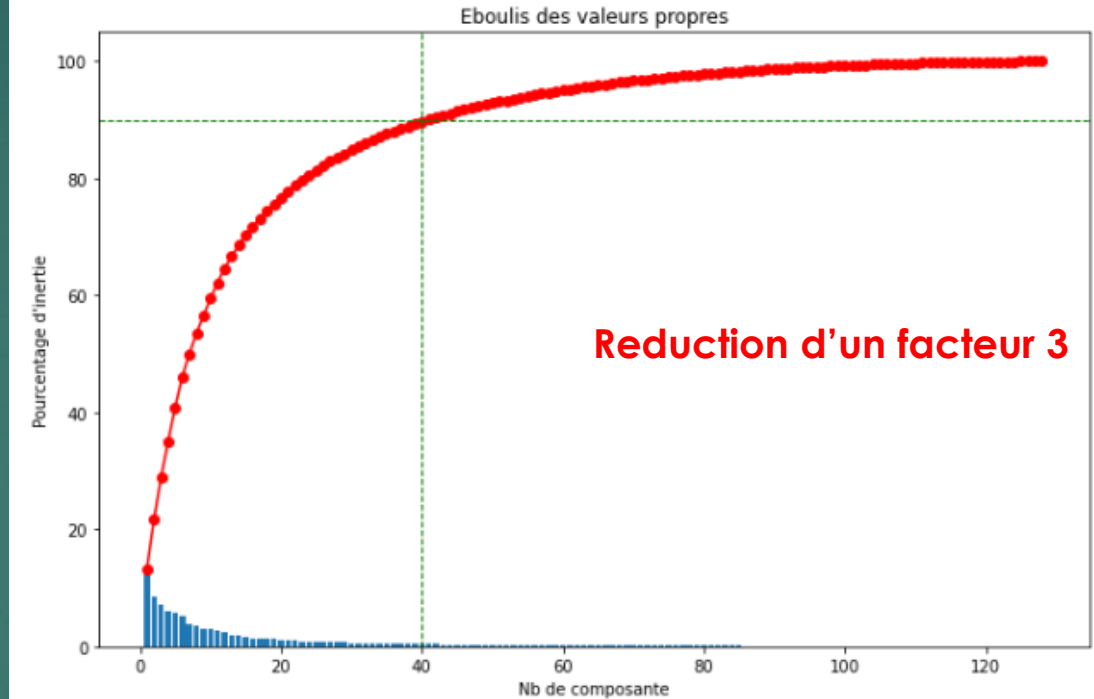
    pca_spark = pyspark.ml.feature.PCA(inputCol="features_vd", outputCol="features_pca", k=min_plans)
    pca = pca_spark.fit(img_vd_df)
    pca_matrix = pca.transform(img_vd_df)

    return pca_matrix

Entrée [16]: pca_matrix = PCA (df_feat,nb_composante)
```



90% de la variance expliquée par 40 composantes



```
pca_matrix.show(1)
```

```
+-----+-----+
| features_pca | img_path |
+-----+-----+
|[277.210683899983...|data_test/apple_6...|
+-----+-----+
only showing top 1 row
```

Etape 6/

Export des données dans le bucket S3

```
Entrée [19]: #Sauvegarde du fichier :  
s3_resource = boto3.resource('s3')  
# Création d'un buffer  
csv_buffer = StringIO()  
# Transformation dans un structure dataframe pandas  
pca_matrix.toPandas().to_csv(csv_buffer)  
# Ecriture du fichier csv dans le bucket s3  
s3_resource.Object('p8-resultats', 'p8_result_with_pca.csv').put(Body=csv_buffer.getvalue())
```

```
Out[19]: {'ResponseMetadata': {'RequestId': '23T4SHA2F86MCJCZ',  
  'HostId': '7Y9/4WcPx6XRyNwBIK5S512xzkEG4MLqgCoJ5CDk6zvKZo06aoGL/jWLNgCqmmeh1MtEpvsejH4=',  
  'HTTPStatusCode': 200,  
  'HTTPHeaders': {'x-amz-id-2': '7Y9/4WcPx6XRyNwBIK5S512xzkEG4MLqgCoJ5CDk6zvKZo06aoGL/jWLNgCqmmeh1MtEpvsejH4=',  
    'x-amz-request-id': '23T4SHA2F86MCJCZ',  
    'date': 'Wed, 14 Dec 2022 15:40:09 GMT',  
    'x-amz-server-side-encryption': 'AES256',  
    'etag': '"da5e152b554f316718385260335d555b"',  
    'server': 'AmazonS3',  
    'content-length': '0'},  
  'RetryAttempts': 0},  
  'ETag': '"da5e152b554f316718385260335d555b"',  
  'ServerSideEncryption': 'AES256'}
```

Conclusion

20

- **Utilisation de PySpark**
- Mise en place d'un environnement Big Data et utilisation des outils pour manipuler les données : **S3 –SAGEMAKER –IAM**
- **SageMaker est un bon outil pour les data scientist :**
 - Rapide et simple à utiliser
 - Pas beaucoup de package à installer
 - Possibilité de changer le type d'instance pour l'adapter à nos travaux
- **Mais le cout est plus élevé comparé à EC2**