



Instituto Federal de Educação, Ciência e Tecnologia
Câmpus Campinas

Professores: Bianca Pedrosa
Samuel Martins

Projeto Interdisciplinar

1. Especificação

O objetivo do projeto interdisciplinar é a aplicação das técnicas de Ciência de Dados aprendidas nas diferentes disciplinas do semestre, a partir de um problema apoiado por um conjunto de dados (dataset). O projeto requer o desenvolvimento de uma solução de aprendizado de máquina cujo treinamento e processamento deverão ser realizados em ambiente de nuvem (AWS ou Google), com processamento distribuído (cluster de computadores), explorando o uso de paralelismo (spark/mapreduce).

Aluno: Omar Hajime Fidelis

Data: 05/12/2021

2. Regras Gerais

- O tema do projeto é de livre escolha e deve ser levantado pelos alunos integrantes do grupo.
- O projeto deverá ser feito em grupos de até 3 (três) pessoas.
- Ao final do semestre, os alunos devem elaborar uma apresentação do projeto, mostrando os resultados das respectivas atividades requeridas por cada uma das disciplinas cursadas.
- Todos os grupos devem apresentar o trabalho. A entrega pelo Moodle é só uma formalidade e não garante avaliação. Os alunos que não participarem da apresentação, a princípio, terão nota 0.
- Cada professor exigirá atividades relativas a conteúdos específicos de sua disciplina.
- As notas das disciplinas serão obtidas separadamente.

Orientações para elaboração do Trabalho Interdisciplinar Big Data + ML

2º semestre 2021 – Prof. Bianca

Detalhamento da parte do trabalho referente à Big Data

Processamento de algoritmos de ML em ambiente distribuído, formado por um cluster de máquinas rodando hadoop map reduce/spark. Estes ambientes podem ser encontrados na AWS e Google, entre outros.

1. Organização do conteúdo do trabalho

Os trabalhos devem ter os seguintes tópicos:

- a) Introdução, que deve ser uma breve contextualização.
- b) Descrição dos dados (fonte, período, volume, formato, amostras dos dados)
- c) Workflow/pipeline de arquitetura
- d) Infraestrutura (Configuração do cluster como nro de nós, tipo de máquinas, permissões)
- e) Análise do Processamento (análise do desempenho das operações ETL, segundo critérios como tempo de processamento, formas intermediárias de armazenamento dos dados (partições), comparação do desempenho com diferentes configurações de cluster)
- f) Funções definidas para processamento distribuído (map, reduce, outras)
- g) Conclusões (limitações e vantagens)
- h) Referências

1. Introdução

1.1. Objetivo

Este projeto tem como objetivo a aplicação das técnicas de Ciência de Dados no conjunto de dados (dataset) de "Wine Quality". A escolha do dataset está relacionado a continuidade e comparação do processo de desenvolvimento do projeto do semestre anterior.

O foco principal deste projeto é o desenvolvimento de uma solução de aprendizado de máquina cujo treinamento e processamento deverão ser realizados em ambiente de nuvem (Amazon Web Service - AWS), com processamento distribuído (cluster de computadores), explorando o uso de paralelismo (spark/mapreduce).

Além o ambiente distribuído, aplicou-se técnicas de machine learning (normalização, pipeline, cross validation), e uma série de algoritmos de classificação para avaliação de performance e resultados.

1.2. Descrição Dataset

A base de dados consolidada é composta por 1599 amostra de vinho vermelho e 4898 amostra de vinho branco, totalizando 6497 amostra e descritas pela coluna "style". O dataset foi criado Paulo Cortez (Univ. Minho), António Cerdeira, Fernando Almeida, Telmo Matos and José Reis em 2009.

Cada linha da base de dados é uma amostra de vinho com suas medições físico-químicas, tipo do vinho (vermelho ou branco) e a nota/avaliação de especialistas.

Neste projeto, consideramos a análise para a classificação do vinho, podendo ser "tinto" ou "branco".

1.3. Detalhamento dos atributos

fixed_acidity = Acidez Fixa: ácido não volátil encontrado no vinho, que é tartárico, málico, cítrico e succínico. Todos esses ácidos são originários das uvas, com exceção do ácido succínico, que é produzido pela levedura durante o processo de fermentação

volatile_acidity = Acidez volátil: os elementos ácidos de um vinho que são gasosos, em vez de líquidos e, portanto, podem ser sentidos como um cheiro, mostrando um aroma, em vez de encontrados no palato

citric_acid = Ácido cítrico: um ácido orgânico fraco, frequentemente usado como conservante natural ou aditivo para alimentos ou bebidas para adicionar um sabor azedo e frescor aos alimentos

residual_sugar = Açúcar residual: a quantidade de açúcar remanescente após a parada da fermentação, é raro encontrar vinhos com menos de 1 grama / litro e vinhos com mais de 45 gramas / litro são considerados doces

chlorides = Cloretos: a quantidade de sal no vinho

free_sulfur_dioxide = Dióxido de Enxofre Livre (SO₂): o SO₂ é usado em todas as fases do processo de vinificação para prevenir a oxidação e o crescimento microbiano. Quantidades excessivas de SO₂ podem inibir a fermentação e causar efeitos sensoriais indesejáveis

total_sulfur_dioxide = Dióxido de Enxofre Total: quantidade das formas livre e ligada de SO₂; em baixas concentrações, SO₂ é principalmente indetectável no vinho, mas em concentrações de SO₂ livre acima de 50 ppm, SO₂ torna-se evidente no nariz e no sabor do vinho

density = Densidade: a densidade é próxima à da água, dependendo da porcentagem de álcool e açúcar

pH: produtores de vinho usam o pH como forma de medir a maturação em relação à acidez. Vinhos de pH baixo terão gosto ácido e crocante, enquanto vinhos de pH mais alto são mais suscetíveis ao crescimento bacteriano. A maioria do pH do vinho está em torno de 3 ou 4

sulphates = Sulfatos: aditivo do vinho que pode contribuir para os níveis de gás de dióxido de enxofre (SO₂), que atua como um antimicrobiano e antioxidante

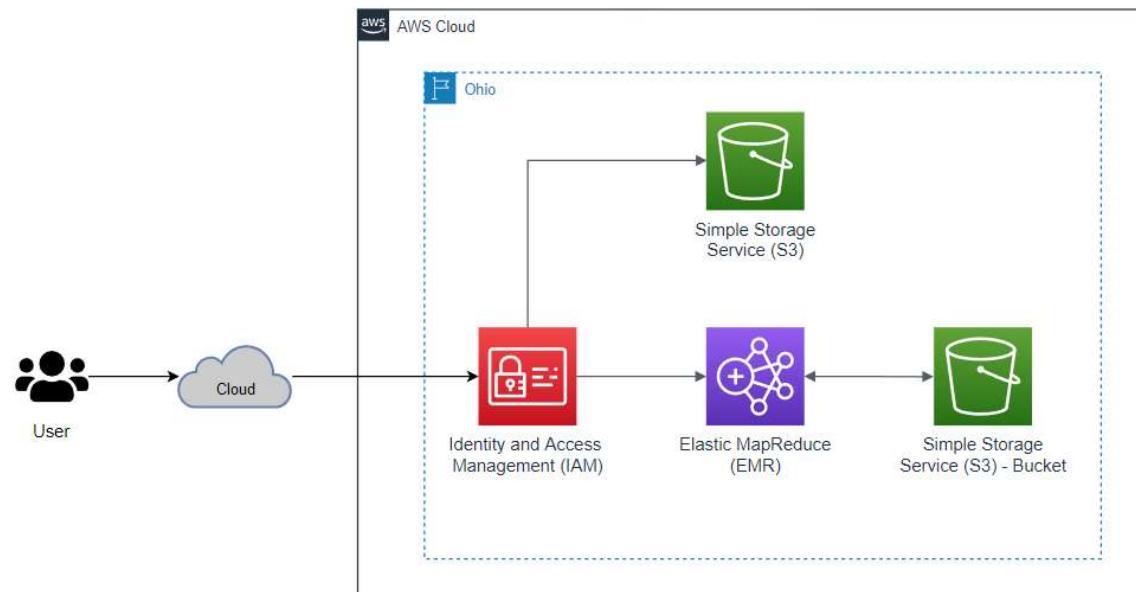
alcohol = Álcool: o teor de álcool do vinho em percentual

quality = Qualidade: notas/avaliações realizadas por no mínimo 3 especialistas em vinhos

style = Tipo: Tipo do Vino (Vermelho ou Branco)

2. Infraestrutura

2.1. Infraestrutura AWS



2.2. AWS Applications

2.2.1. Apache Spark

Apache Spark is a distributed processing framework and programming model that helps you do machine learning, stream processing, or graph analytics using Amazon EMR clusters. Similar to Apache Hadoop, Spark is an open-source, distributed processing system commonly used for big data workloads. However, Spark has several notable differences from Hadoop MapReduce. Spark has an optimized directed acyclic graph (DAG) execution engine and actively caches data in-memory, which can boost performance, especially for certain algorithms and interactive queries.

Spark version information for emr-5.33.0		
Amazon EMR Release Label	Spark Version	Components Installed With Spark
emr-5.33.0	Spark 2.4.7	aws-sagemaker-spark-sdk, emrfs, emr-goodies, emr-ddb, emr-s3-select, hadoop-client, hadoop-hdfs-datanode, hadoop-hdfs-library, hadoop-hdfs-namenode, hadoop-https-server, hadoop-kms-server, hadoop-yarn-nodemanager, hadoop-yarn-resourcemanager, hadoop-yarn-timeline-server, hudi, hudi-spark, livy-server, nginx, r, spark-client, spark-history-server, spark-on-yarn, spark-yarn-slave

2.2.2. Apache Hive

Hive is an open-source, data warehouse, and analytic package that runs on top of a Hadoop cluster. Hive scripts use an SQL-like language called Hive QL (query language) that abstracts programming models and supports typical data warehouse interactions. Hive enables you to avoid the complexities of writing Tez jobs based on directed acyclic graphs (DAGs) or MapReduce programs in a lower level computer language, such as Java.

Hive version information for emr-6.4.0		
Amazon EMR Release Label	Hive Version	Components Installed With Hive
emr-6.4.0	Hive 3.1.2	emrfs, emr-ddb, emr-goodies, emr-kinesis, emr-s3-dist-cp, emr-s3-select, hadoop-client, hadoop-mapred, hadoop-hdfs-datanode, hadoop-hdfs-library, hadoop-hdfs-namenode, hadoop-https-server, hadoop-kms-server, hadoop-yarn-nodemanager, hadoop-yarn-resourcemanager, hadoop-yarn-timeline-server, hive-client, hive-hbase, hcatalog-server, hive-server2, hudi, mariadb-server, tez-on-yarn, zookeeper-client, zookeeper-server

2.2.3. Apache Livy

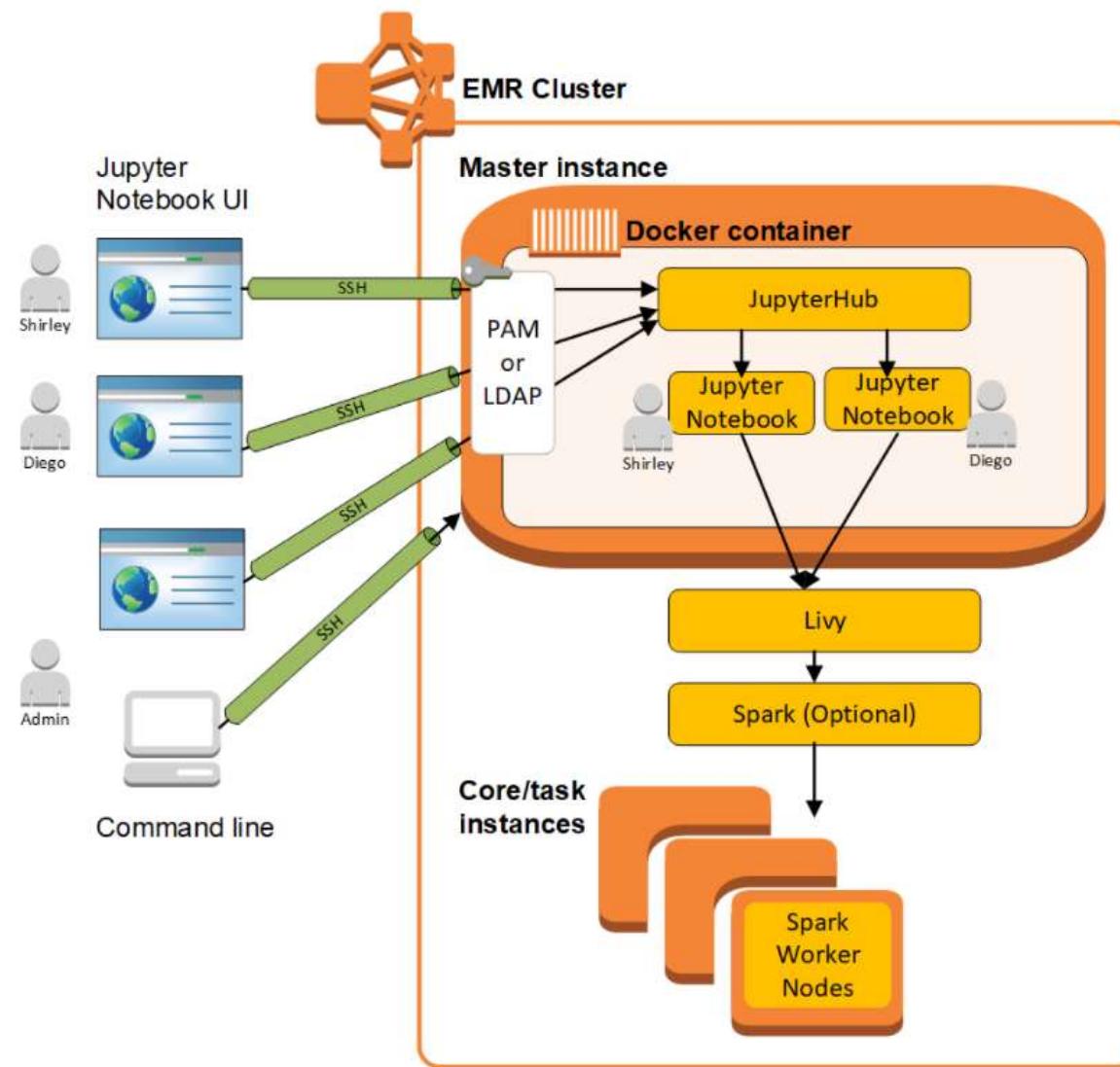
Livy enables interaction over a REST interface with an EMR cluster running Spark. You can use the REST interface or an RPC client library to submit Spark jobs or snippets of Spark code, retrieve results synchronously or asynchronously, and manage Spark Context. For more information, see the Apache Livy website. Livy is included in Amazon EMR release version 5.9.0 and later.

Livy version information for emr-6.4.0		
Amazon EMR Release Label	Livy Version	Components Installed With Livy
emr-6.4.0	Livy 0.7.1	aws-sagemaker-spark-sdk, emrfs, emr-goodies, emr-ddb, hadoop-client, hadoop-hdfs-datanode, hadoop-hdfs-library, hadoop-hdfs-namenode, hadoop-kms-server, hadoop-yarn-nodemanager, hadoop-yarn-resourcemanager, hadoop-yarn-timeline-server, r, spark-client, spark-history-server, spark-on-yarn, spark-yarn-slave, livy-server, nginx

2.2.4. Amazon EMR Notebook based on Jupyter Notebook

EMR Notebooks is a Jupyter Notebook environment built in to the Amazon EMR console that allows you to quickly create Jupyter notebooks, attach them to Spark clusters, and then open the Jupyter Notebook editor in the console to remotely run queries and code. An EMR notebook is saved in Amazon S3 independently from clusters for durable storage, quick access, and flexibility. You can have multiple notebooks open, attach multiple notebooks to a single cluster, and re-use a notebook on different clusters.

2.2.4. Amazon EMR Notebook Arquitetura



2.3. Create EMR Cluster

Lista de aplicações do Cluster

Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

Software Configuration

Release: emr-5.33.1

Hadoop 2.10.1 Zeppelin 0.9.0 Livy 0.7.0
 JupyterHub 1.1.0 Tez 0.9.2 Flink 1.12.1
 Ganglia 3.7.2 HBase 1.4.13 Pig 0.17.0
 Hive 2.3.7 Presto 0.245.1 ZooKeeper 3.4.14
 JupyterEnterpriseGateway 2.1.0 MXNet 1.7.0 Sqoop 1.4.7
 Mahout 0.13.0 Hue 4.9.0 Phoenix 4.14.3
 Oozie 5.2.0 Spark 2.4.7 HCatalog 2.3.7
 TensorFlow 2.4.1

Multiple master nodes (optional)

Use multiple master nodes to improve cluster availability. [Learn more](#)

AWS Glue Data Catalog settings (optional)

Use for Hive table metadata [Learn more](#)
 Use for Spark table metadata [Learn more](#)

Edit software settings [Edit](#)

Enter configuration Load JSON from S3

```
classification=config-file-name.properties=[myKey1=myValue1,myKey2=myValue2]
```

Steps (optional)

A step is a unit of work you submit to the cluster. For instance, a step might contain one or more Hadoop or Spark jobs. You can also submit additional steps to a cluster after it is running. [Learn more](#)

Concurrency: Run multiple steps at the same time to improve cluster utilization

After last step completes: Clusters enters waiting state Cluster auto-terminates

Step type: [Select a step](#) [Add step](#)

Feedback English (US) [▼](#)

© 2021, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Hardware Configuration

Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

Hardware Configuration

Specify the networking and hardware configuration for your cluster. Request Spot Instances (unused EC2 capacity) to save money.

Cluster Composition

Specify the configuration of the master, core and task nodes as an instances group or instance fleet. This choice applies to all nodes for the lifetime of the cluster. Instance fleets and instance groups cannot coexist in a cluster. [see this topic](#)

Instance group configuration

- Uniform instance groups

Specify a single instance type and purchasing option for each node type.
- Instance fleets

Specify target capacity and how Amazon EMR fulfills it for each node type. Mix instance types and purchasing options. [Learn more](#)

Networking

Use a Virtual Private Cloud (VPC) to process sensitive data or connect to a private network. Launch the cluster into a VPC with a public, private or shared subnet. Subnets may be associated with an AWS Outpost or AWS Local Zone.

Network: [vpc-99851ef2 \(172.31.0.0/16\) \(default\)](#) | [Create a VPC](#)

EC2 Subnet: [subnet-05798078 | Default in us-east-2b](#)

Cluster Nodes and Instances

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Console options for automatic scaling have changed. [Learn more](#)

Node type	Instance type	Instance count	Purchasing option
-----------	---------------	----------------	-------------------

Feedback English (US) © 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Hardware Description

Cluster Nodes and Instances

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Node type	Instance type	Instance count	Purchasing option
Master	m5.xlarge	1 Instances	<input checked="" type="radio"/> On-demand <small>On-demand</small> <input type="radio"/> Spot <small>Spot</small> <small>Use on-demand as max price</small>
Master Instance Group	4 vCore, 16 GiB memory, EBS only storage EBS Storage: 64 GiB		
Core	m5.xlarge	3 Instances	<input checked="" type="radio"/> On-demand <small>On-demand</small> <input type="radio"/> Spot <small>Spot</small> <small>Use on-demand as max price</small>
Core - 2	4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB		
Add configuration settings + Add task instance group			

Total core and task units: 3 Total units

Cluster scaling

Adjust the number of Amazon EC2 instances available to an EMR cluster via EMR-managed scaling or a custom automatic scaling policy. [Learn more](#)

Cluster scaling Enable Cluster Scaling

Use EMR-managed scaling
 Create a custom automatic scaling policy

EMR-managed scaling

EMR will automatically increase and decrease the number of instances in core and task nodes based on workload. Set a minimum and maximum limit of the number of instances for the cluster nodes. Master nodes do not scale.

Core and task units

Minimum:	3
Maximum:	6
On-demand limit:	6
Maximum Core Node:	6

Feedback English (US) [▼](#) © 2021, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Habilitando Cluster Scaling

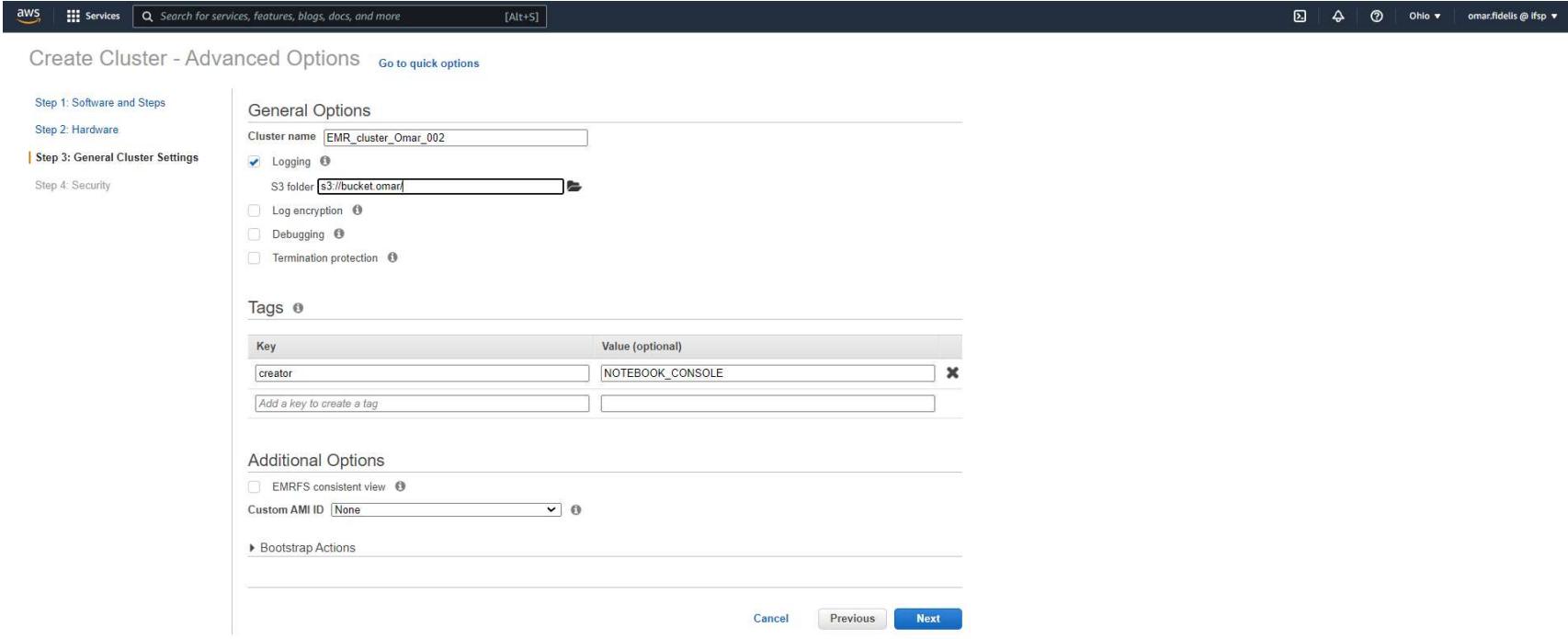
The screenshot shows the 'Cluster configuration' step of an AWS EMR cluster setup. The 'Core and task units' section shows '3 Total units'. The 'Cluster scaling' section is expanded, showing 'Enable Cluster Scaling' (checked) and 'Use EMR-managed scaling' (selected). The 'EMR-managed scaling' description states that EMR will automatically increase and decrease the number of instances based on workload. The 'Core and task units' configuration shows 'Minimum: 3', 'Maximum: 6', 'On-demand limit: 6', and 'Maximum Core Node: 6'. The 'Auto-termination' section shows 'Enable auto-termination' (checked) and 'Terminate cluster when it is idle after 0 hours 10 minutes'. The 'EBS Root Volume' section shows 'Root device EBS volume size: 20 GB'. At the bottom, there are 'Cancel', 'Previous', and 'Next' buttons, along with a copyright notice for Amazon Web Services and links for Feedback, English (US), Privacy, Terms, and Cookie preferences.

Aumento hardware configuration

The screenshot shows the AWS EMR Cluster Configuration page. At the top, it displays 'Total core and task units' and '1 Total units'. The 'Cluster scaling' section is active, with 'Enable Cluster Scaling' checked and 'Use EMR-managed scaling' selected. Below this, the 'EMR-managed scaling' description states that EMR will automatically increase and decrease the number of instances in core and task nodes based on workload. The 'Core and task units' section shows settings for 'Minimum' (2), 'Maximum' (10), 'On-demand limit' (10), and 'Maximum Core Node' (10). The 'Auto-termination' section allows setting a time for the cluster to terminate after it becomes idle, with 'Enable auto-termination' checked and 'Terminate cluster when it is idle after 0 hours 10 minutes'. The 'EBS Root Volume' section specifies a root device volume size of 10 GiB. At the bottom, there are 'Cancel', 'Previous', and 'Next' buttons, along with copyright and legal links.

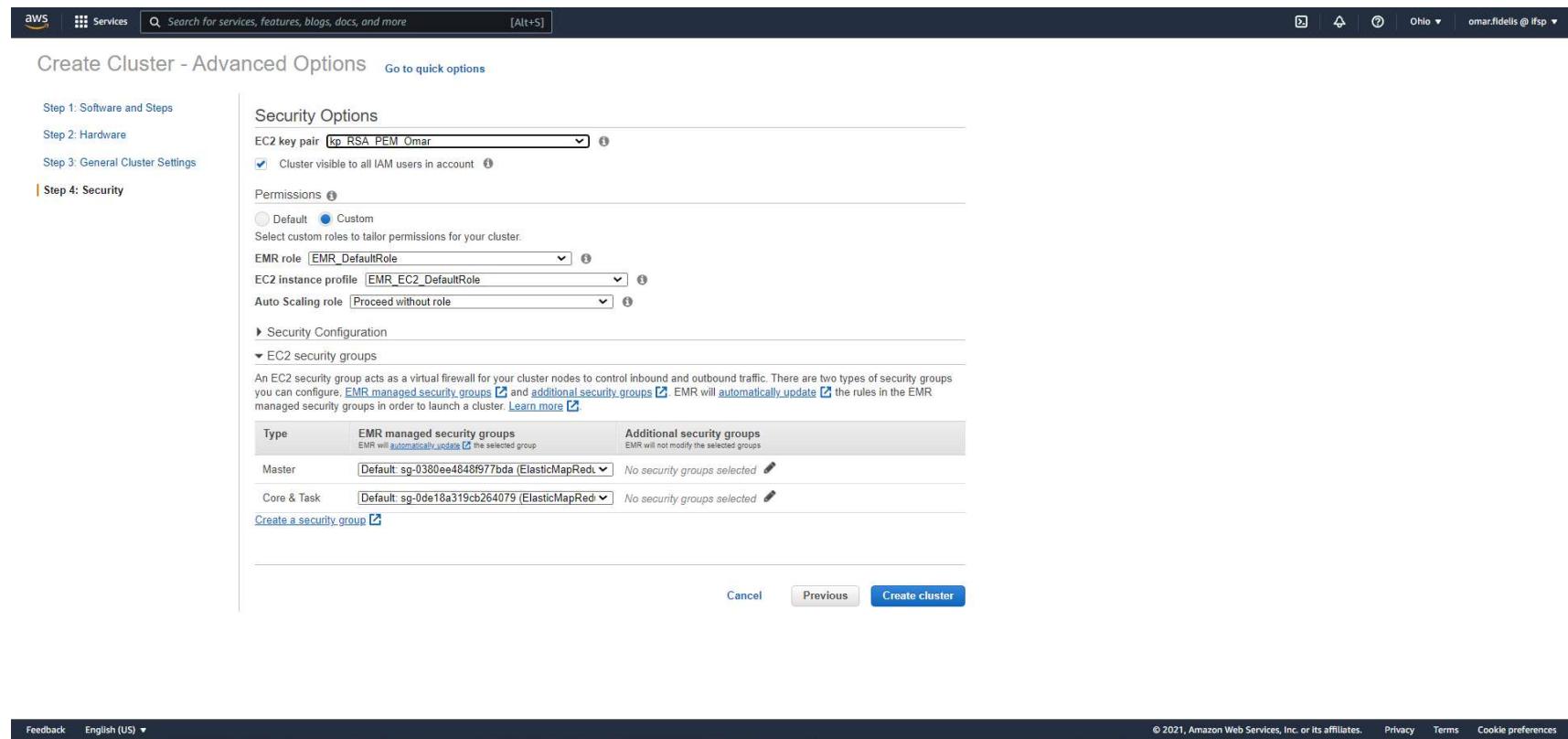
Definição de Bucket de LOG

Screenshot of the AWS EMR 'Create Cluster - Advanced Options' step 3: General Cluster Settings. The 'Logging' checkbox is checked, and the S3 folder is set to 's3://bucket.omar/'. The 'Tags' section contains a single tag 'creator' with value 'NOTEBOOK_CONSOLE'. The 'Additional Options' section shows 'EMRFS consistent view' unchecked and 'Custom AMI ID' set to 'None'. The 'Bootstrap Actions' section is empty. Navigation buttons at the bottom are 'Cancel', 'Previous', and a large 'Next' button.



Feedback English (US) ▾ © 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Configuração padrão (key, role, profile, etc.)



Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

Security Options

EC2 key pair **Kp RSA PEM Omar**

Cluster visible to all IAM users in account [?](#)

Permissions

Default Custom

Select custom roles to tailor permissions for your cluster.

EMR role **EMR_DefaultRole**

EC2 Instance profile **EMR_EC2_DefaultRole**

Auto Scaling role **Proceed without role**

EC2 security groups

An EC2 security group acts as a virtual firewall for your cluster nodes to control inbound and outbound traffic. There are two types of security groups you can configure: [EMR managed security groups](#) and [additional security groups](#). EMR will [automatically update](#) the rules in the EMR managed security groups in order to launch a cluster. [Learn more](#)

Type	EMR managed security groups	Additional security groups
Master	Default: sg-0380ee4848f977bda (ElasticMapRed) Edit No security groups selected Edit	EMR will automatically update the selected group
Core & Task	Default: sg-0de18a319cb264079 (ElasticMapRed) Edit No security groups selected Edit	EMR will not modify the selected groups

[Create a security group](#)

Cancel Previous **Create cluster**

Feedback English (US) [▼](#)

© 2021, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Running server

Cluster: EMR_cluster_Omar_002 Waiting Cluster ready to run steps.

Summary

ID: j-11RDB040LSPLE
Creation date: 2021-11-28 12:59 (UTC-3)
Elapsed time: 10 minutes
After last step completes: Cluster waits
Termination protection: Off [Change](#)
Tags: creator = NOTEBOOK_CONSOLE [View All / Edit](#)

Master public DNS: ec2-3-23-94-223.us-east-2.compute.amazonaws.com [Copy](#)
[Connect to the Master Node Using SSH](#)

Application user interfaces

Persistent user interfaces [Edit](#): Spark history server, YARN timeline server, Tez UI

On-cluster user interfaces [Edit](#): Not Enabled [Enable an SSH Connection](#)

Network and hardware

Availability zone: us-east-2b
Subnet ID: [subnet-05798078](#) [Edit](#)
Master: [Running](#) 1 m5.xlarge
Core: --
Task: --
Cluster scaling: Not enabled
Auto-termination: --

Security and access

Key name: --
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All [Change](#)
Security groups for Master: [sg-0380ee4848f977bda](#) [Edit](#) (ElasticMapReduce-master)
Security groups for Core & Task: [sg-0de18a319cb264079](#) [Edit](#) (ElasticMapReduce-task: slave)

Running Jupyter Notebook

The screenshot shows the AWS Management Console interface for the Amazon EMR service. The left sidebar lists various EMR-related options: Amazon EMR, EMR Studio, EMR on EC2, Clusters, Notebooks (which is selected and highlighted in orange), Git repositories, Security configurations, Block public access, VPC subnets, Events, EMR on EKS, and Virtual clusters. Below this is a 'Help' section and a 'What's new' link.

The main content area displays a notebook named 'PI_Omar' in a 'Ready' state. The notebook's ID is e-3UBAAQ80F5GR21GUCMA770VVS. It has a description of '--'. The notebook was last modified 0 seconds ago by the user 'omar.fidelis'. It was created on 2021-11-22 21:27 (UTC-3) by the user 'omar.fidelis'. The Service IAM role is EMR_Notebooks_DefaultRole. The notebook is associated with two security groups: sg-0fee1b6e9eee7a001 (master instance) and sg-011bcd53cc689e26b (notebook instance). The notebook tags include creatorUserId = AIDAQEQQ7R7JUO2YK2JQ6Q. The notebook is located at s3://bucket.omar/.

The 'Cluster' section shows the cluster 'EMR_cluster_Omar_002' with a cluster ID of j-11RDB040LSPLE. The cluster status is 'Waiting' (Cluster ready to run steps). Cluster tags include creator = NOTEBOOK_CONSOLE. Step logs are located at s3://aws-logs-009729866344-us-east-2/elasticmapreduce/.

The 'Git repositories' section indicates that the repository can be linked to the notebook once the notebook is ready. It includes 'Link new repository' and 'Unlink repository' buttons. A table lists repositories with columns: Repository name, URL, Branch, Link status, and Failure reason. The table is currently empty.

At the bottom, there are links for 'Feedback', 'English (US) ▾', and copyright information: © 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

2.4. Hardware Configuration

Summary Configuration

Cluster: EMR_cluster_Omar_002 Waiting Cluster ready to run steps.

Summary

ID: J-11RDB040LSPLE
 Creation date: 2021-11-28 12:59 (UTC-3)
 Elapsed time: 2 hours
 After last step completes: Cluster waits
 Termination protection: Off [Change](#)
 Master public DNS: [ec2-3-23-94-223.us-east-2.compute.amazonaws.com](#) [View All / Edit](#)
 Connect to the Master Node Using SSH

Application user interfaces

Persistent user interfaces: [Spark history server](#), [YARN timeline server](#), [Tez UI](#)
 On-cluster user interfaces: Not Enabled [Enable an SSH Connection](#)

Configuration details

Release label: emr-5.33.1
 Hadoop distribution: Amazon 2.10.1
 Applications: Spark 2.4.7, Livy 0.7.0, Hive 2.3.7, JupyterEnterpriseGateway 2.1.0
 Log URI: [s3://aws-logs-009729866344-us-east-2/elasticmapreduce/](#) [View Log](#)

EMRFS consistent view: Disabled
 Custom AMI ID: --

Network and hardware

Availability zone: us-east-2b
 Subnet ID: [subnet-05798078](#) [View](#)
 Master: [Running](#) 1 m5.xlarge
 Core: --
 Task: --
 Cluster scaling: Not enabled
 Auto-termination: --

Security and access

Key name: --
 EC2 instance profile: EMR_EC2_DefaultRole
 EMR role: EMR_DefaultRole
 Visible to all users: All [Change](#)
 Security groups for Master: [sg-0380ee4848f977bda](#) (ElasticMapReduce-master)
 Security groups for Core & Task: [sg-0de18a319cb264079](#) (ElasticMapReduce-Task: slave)

Hardware Configuration

Cluster: EMR_cluster_Omar_002 Waiting Cluster ready to run steps.

Summary **Application user interfaces** **Monitoring** **Hardware** **Configurations** **Events** **Steps** **Bootstrap actions**

Add task instance group

Instance groups

Filter: 1 instance group (all loaded) C

ID	Status	Node type & name	Instance type	Instance count	Purchasing option
ig-1BK80S8ADEPZZ	Running	MASTER Master Instance Group	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 64 GiB	1 Instances	On-demand i

Cluster Scaling Policy Edit

No scaling enabled

Auto-termination Edit

Select a time to have the cluster terminate after the cluster becomes idle. Choose a minimum of 1 minute or a max of 24 hours. [Learn more](#) i

Auto-termination:

Feedback English (US) © 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Application Configuration

Cluster: EMR_cluster_Omar_002 Waiting Cluster ready to run steps.

Persistent application user interfaces

Applications installed on the Amazon EMR cluster publish user interfaces (UI) as web sites to monitor cluster activity. Persistent UI logs are available for 30 days after an application ends. Persistent UI don't require SSH tunneling. They are hosted off of the cluster.

Application user interface

- [YARN timeline server](#)
- [Spark history server](#)
- [Tez UI](#)

On-cluster application user interfaces

On-cluster UI are available only while clusters are running. Because they are hosted on the master node, on-cluster UI require a connection via SSH tunneling. Set up SSH tunneling before accessing these application UI. [Learn more](#)

Application	User Interface URL	Status
HDFS Name Node	http://ec2-3-23-94-223.us-east-2.compute.amazonaws.com:50070/	SSH tunnel not enabled
Spark History Server	http://ec2-3-23-94-223.us-east-2.compute.amazonaws.com:18080/	SSH tunnel not enabled
Livy	http://ec2-3-23-94-223.us-east-2.compute.amazonaws.com:8998/	SSH tunnel not enabled
Resource Manager	http://ec2-3-23-94-223.us-east-2.compute.amazonaws.com:8088/	SSH tunnel not enabled

The following table lists web interfaces you can view on the task nodes:

Application	User interface URL
HDFS Data Node	http://ec2-000-000-000-000.compute-1.amazonaws.com:50075/
Node Manager	http://ec2-000-000-000-000.compute-1.amazonaws.com:8042/

High-level application history

Amazon EMR collects information from YARN applications on your cluster and keeps a summary of historical information for seven days after applications have completed. [Learn more](#)

YARN applications (9)

Application ID	Type	Action	Status	Start time (UTC-3)	Duration	Finish time (UTC-3)	User
...

Filter: [All applications](#) [Filter applications ...](#) 9 applications (all loaded)

© 2021, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Spark Sessions

On-cluster application user interfaces

On-cluster UI are available only while clusters are running. Because they are hosted on the master node, on-cluster UI require a connection via SSH tunneling. Set up SSH tunneling before accessing these application UI. [Learn more](#)

Application	User Interface URL	Status
HDFS Name Node	http://ec2-3-23-94-223.us-east-2.compute.amazonaws.com:50070/	SSH tunnel not enabled
Spark History Server	http://ec2-3-23-94-223.us-east-2.compute.amazonaws.com:18080/	SSH tunnel not enabled
Livy	http://ec2-3-23-94-223.us-east-2.compute.amazonaws.com:8998/	SSH tunnel not enabled
Resource Manager	http://ec2-3-23-94-223.us-east-2.compute.amazonaws.com:8088/	SSH tunnel not enabled

The following table lists web interfaces you can view on the task nodes:

Application	User interface URL
HDFS Data Node	http://ec2-000-000-000-000.compute-1.amazonaws.com:50075/
Node Manager	http://ec2-000-000-000-000.compute-1.amazonaws.com:8042/

High-level application history

Amazon EMR collects information from YARN applications on your cluster and keeps a summary of historical information for seven days after applications have completed. [Learn more](#)

YARN applications (9)

Application ID	Type	Action	Status	Start time (UTC-3)	Duration	Finish time (UTC-3)	User
▶ application_1638115520361_0009	Spark	livy-session-8	Running	2021-11-28 14:20 (UTC-3)	37 min		livy
▶ application_1638115520361_0008	Spark	livy-session-7	Succeeded	2021-11-28 14:18 (UTC-3)	18 s	2021-11-28 14:18 (UTC-3)	livy
▶ application_1638115520361_0007	Spark	livy-session-6	Succeeded	2021-11-28 14:16 (UTC-3)	1.3 min	2021-11-28 14:17 (UTC-3)	livy
▶ application_1638115520361_0006	Spark	livy-session-5	Succeeded	2021-11-28 14:16 (UTC-3)	1.3 min	2021-11-28 14:17 (UTC-3)	livy
▶ application_1638115520361_0005	Spark	livy-session-4	Succeeded	2021-11-28 14:04 (UTC-3)	5.3 min	2021-11-28 14:10 (UTC-3)	livy
▶ application_1638115520361_0004	Spark	livy-session-3	Succeeded	2021-11-28 13:47 (UTC-3)	7.3 min	2021-11-28 13:54 (UTC-3)	livy
▶ application_1638115520361_0003	Spark	livy-session-2	Succeeded	2021-11-28 13:40 (UTC-3)	6.1 min	2021-11-28 13:46 (UTC-3)	livy
▶ application_1638115520361_0002	Spark	livy-session-1	Succeeded	2021-11-28 13:29 (UTC-3)	10 min	2021-11-28 13:40 (UTC-3)	livy
▶ application_1638115520361_0001	Spark	livy-session-0	Succeeded	2021-11-28 13:21 (UTC-3)	8.0 min	2021-11-28 13:29 (UTC-3)	livy

Feedback English (US) ▾

© 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

3. Preparação do ambiente

In [1]: *# Import Library*

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI
0	application_1638144682166_0001	pyspark	idle	Link (http://ip-172-31-23-153.us-east-2.compute.internal:20888/proxy/application_1638144682166_0001/) , 2.compute.internal:8042/node/

SparkSession available as 'spark'.

In [2]: *# Start Spark Session*

```
spark = SparkSession \
    .builder \
    .getOrCreate()
```

In [3]: *# Data Structure*

```
schema = StructType([
    StructField("Acidez_Fixa", FloatType(), True), \
    StructField("Acidez_Volatil", FloatType(), True), \
    StructField("Acido_Citrico", FloatType(), True), \
    StructField("Acucar_Residual", FloatType(), True), \
    StructField("Cloreto_Sodio", FloatType(), True), \
    StructField("Dioxido_Enxofre_Livre", FloatType(), True), \
    StructField("Dioxido_Enxofre_Total", FloatType(), True), \
    StructField("Densidade", FloatType(), True), \
    StructField("pH", FloatType(), True), \
    StructField("Sulfato", FloatType(), True), \
    StructField("Alcool", FloatType(), True), \
    StructField("Qualidade", FloatType(), True), \
    StructField("Tipo", StringType(), True)])
```

3.1. Preparação dataset

```
In [4]: # Import Dataset from S3 Bucket
df = spark.read.schema(schema).option("header","true").csv(r"s3n://bucket.omar/dataset/wine_dataset.csv")
```

```
In [5]: # Overview
df.show(5)
```

▶ Spark Job Progress

	Acidez_Fixa	Acidez_Volatil	Acido_Citrico	Acucar_Residual	Cloreto_Sodio	Dioxido_Enxofre_Livre	Dioxido_Enxofre_Total	Densidade	pH	Sulfato	Alcool	Qualidade	Tipo
34.0	7.4	0.9978	3.51	0.56	9.4	5.0	red	1.9	0.076				11.0
67.0	7.8	0.9968	3.2	0.68	9.8	5.0	red	2.6	0.098				25.0
54.0	7.8	0.997	3.26	0.65	9.8	5.0	red	0.04	0.092				15.0
60.0	11.2	0.998	3.16	0.28	9.8	6.0	red	0.56	0.075				17.0
34.0	7.4	0.9978	3.51	0.56	9.4	5.0	red	0.0	0.076				11.0

only showing top 5 rows

In [6]: *# Staset Schema*
df.printSchema()

```
root
|-- Acidez_Fixa: float (nullable = true)
|-- Acidez_Volatil: float (nullable = true)
|-- Acido_Citrico: float (nullable = true)
|-- Acucar_Residual: float (nullable = true)
|-- Cloreto_Sodio: float (nullable = true)
|-- Dioxido_Enxofre_Livre: float (nullable = true)
|-- Dioxido_Enxofre_Total: float (nullable = true)
|-- Densidade: float (nullable = true)
|-- pH: float (nullable = true)
|-- Sulfato: float (nullable = true)
|-- Alcool: float (nullable = true)
|-- Qualidade: float (nullable = true)
|-- Tipo: string (nullable = true)
```

```
In [7]: # No missing values
df.select(df.columns).describe().show()
```

► Spark Job Progress

summary	Acidez_Fixa	Acidez_Volatil	Acido_Citrico	Acucar_Residual	Cloreto_Sodio	Dioxo_Sulfato
ido_Enxofre_Livre	Dioxido_Enxofre_Total	Densidade	pH			
Alcool	Qualidade	Tipo				
count	6497	6497	6497	6497	6497	6497
6497	6497	6497	6497	6497	6497	6497
6497	6497					
mean	7.2153070684174345	0.3396659997464309	0.31863321590700183	5.443235335688381	0.05603386190293733	3
0.525319378174544	115.7445744189626	0.9946966336675106	3.218500847887659	0.5312682763074607	10.4918008	
13027597	5.818377712790519	null				
stddev	1.2964337581432526	0.16463647361833078	0.14531786519512516	4.757803743705875	0.03503360134013471	
17.74939977200251	56.52185452263024	0.002998673927059...	0.16078720153691367	0.1488058737693289	1.19271174	
88301993	0.8732552715311255	null				
min	3.8	0.08	0.0	0.6	0.009	
1.0	6.0	0.98711	2.72	0.22	8.0	
3.0	red					
max	15.9	1.58	1.66	65.8	0.611	
289.0	440.0	1.03898	4.01	2.0	14.9	
9.0	white					

```
In [8]: # Transform categorical attribute ("Tipo") to binary attribute
df = df.withColumn('label', col('Tipo') == 'red')
df = df.drop('Tipo')
df.show(5)
```

▶ Spark Job Progress

```
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+
|Acidez_Fixa|Acidez_Volatil|Acido_Citrico|Acucar_Residual|Cloreto_Sodio|Dioxido_Enxofre_Livre|Dioxido_Enxofre_
Total|Densidade| pH|Sulfato|Alcool|Qualidade|label|
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
|    7.4|     0.7|     0.0|      1.9|    0.076|     11.0|
34.0| 0.9978| 3.51|  0.56|    9.4|    5.0| true|
|    7.8|     0.88|     0.0|      2.6|    0.098|     25.0|
67.0| 0.9968|  3.2|  0.68|    9.8|    5.0| true|
|    7.8|     0.76|     0.04|     2.3|    0.092|     15.0|
54.0| 0.997| 3.26|  0.65|    9.8|    5.0| true|
|   11.2|     0.28|     0.56|     1.9|    0.075|     17.0|
60.0| 0.998| 3.16|  0.58|    9.8|    6.0| true|
|    7.4|     0.7|     0.0|      1.9|    0.076|     11.0|
34.0| 0.9978| 3.51|  0.56|    9.4|    5.0| true|
+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
In [9]: # Cast bynary attribute to numerical attribute named "Label" (standard)
from pyspark.sql.types import IntegerType
df = df.withColumn("label", col("label").cast(IntegerType()))
df.printSchema()
```

```
root
|-- Acidez_Fixa: float (nullable = true)
|-- Acidez_Volatil: float (nullable = true)
|-- Acido_Citrico: float (nullable = true)
|-- Acucar_Residual: float (nullable = true)
|-- Cloreto_Sodio: float (nullable = true)
|-- Dioxido_Enxofre_Livre: float (nullable = true)
|-- Dioxido_Enxofre_Total: float (nullable = true)
|-- Densidade: float (nullable = true)
|-- pH: float (nullable = true)
|-- Sulfato: float (nullable = true)
|-- Alcool: float (nullable = true)
|-- Qualidade: float (nullable = true)
|-- label: integer (nullable = true)
```

4. Processamento (Machine Learning)

```
In [10]: # Import Libraries - Machine Learning
from pyspark.ml.feature import VectorAssembler, Normalizer, StandardScaler
from pyspark.ml.classification import LogisticRegression, NaiveBayes, LinearSVC, RandomForestClassifier, GBTClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClassificationEvaluator
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
```

```
In [11]: # Split dataset (training 80% and test 20%)
training, test = df.randomSplit([0.8, 0.2])
```

```
In [12]: # Create Assembler
assembler = VectorAssembler(inputCols=['Acidez_Fixa', 'Acidez_Volatil', 'Acido_Citrico', 'Acucar_Residual', \
                                         'Cloreto_Sodio', 'Dioxido_Enxofre_Livre', 'Dioxido_Enxofre_Total', \
                                         'Densidade', 'pH', 'Sulfato', 'Alcool', 'Qualidade'], \
                                         outputCol="inputFeatures")
```

```
In [13]: # Standard Scaler (Normalize)
scaler = Normalizer(inputCol = "inputFeatures", outputCol = "features")
```

4.1. Logistic Regression

```
In [14]: # Model Creation
lr = LogisticRegression()
```

```
In [15]: # Creating Pipeline
pipeline_lr = Pipeline(stages = [assembler, scaler, lr])
```

```
In [16]: # Parameter
paramgrid_lr = ParamGridBuilder().addGrid(lr.regParam, [0.0, 0.1]).addGrid(lr.maxIter, [10]).build()
```

```
In [17]: # Evaluator
evaluator_lr = MulticlassClassificationEvaluator(metricName = "accuracy")
```

```
In [18]: # Cross Validation
crossval_lr = CrossValidator(estimator = pipeline_lr, estimatorParamMaps = paramgrid_lr, \
                               evaluator = evaluator_lr, numFolds = 3, seed=10)
```

In [19]: *# Training and evaluation*

```
cvModel_lr = crossval_lr.fit(training)
evaluator_lr.evaluate(cvModel_lr.transform(test))
```

▶ Spark Job Progress

```
0.9805295950155763
```

In [20]: *# Evaluator*

```
evaluator_lr = MulticlassClassificationEvaluator(metricName = "f1")
```

In [21]: *# Cross Validation*

```
crossval_lr = CrossValidator(estimator = pipeline_lr, estimatorParamMaps = paramgrid_lr, \
                             evaluator = evaluator_lr, numFolds = 3, seed=10)
```

In [22]: *# Training and evaluation*

```
cvModel_lr = crossval_lr.fit(training)
evaluator_lr.evaluate(cvModel_lr.transform(test))
```

▶ Spark Job Progress

```
0.9804374227910806
```

4.2. Naive Bayes

In [23]: *# Model Creation*

```
nb = NaiveBayes()
```

In [24]: *# Creating Pipeline*

```
pipeline_nb = Pipeline(stages = [assembler, scaler, nb])
```

```
In [25]: # Parameter
paramgrid_nb = ParamGridBuilder().build()
```

```
In [26]: # Evaluator
evaluator_nb = MulticlassClassificationEvaluator(metricName = "accuracy")
```

```
In [27]: # Cross Validation
crossval_nb = CrossValidator(estimator = pipeline_nb, estimatorParamMaps = paramgrid_nb, \
                           evaluator = evaluator_nb, numFolds = 3, seed=10)
```

```
In [28]: # Training and evaluation
cvModel_nb = crossval_nb.fit(training)
evaluator_nb.evaluate(cvModel_nb.transform(test))
```

▶ Spark Job Progress

0.7725856697819314

```
In [29]: # Evaluator
evaluator_nb = MulticlassClassificationEvaluator(metricName = "f1")
```

```
In [30]: # Cross Validation
crossval_nb = CrossValidator(estimator = pipeline_nb, estimatorParamMaps = paramgrid_nb, \
                           evaluator = evaluator_nb, numFolds = 3, seed=10)
```

```
In [31]: # Training and evaluation
cvModel_nb = crossval_nb.fit(training)
evaluator_nb.evaluate(cvModel_nb.transform(test))
```

▶ Spark Job Progress

0.6937899278494694

4.3. Linear Support Vector Machine

```
In [32]: # Model Creation
lsvc = LinearSVC()
```

```
In [33]: # Creating Pipeline
pipeline_lsvc = Pipeline(stages = [assembler, scaler, lsvc])
```

```
In [34]: # Parameter
paramgrid_lsvc = ParamGridBuilder().build()
```

```
In [35]: # Evaluator
evaluator_lsvc = MulticlassClassificationEvaluator(metricName = "accuracy")
```

```
In [36]: # Cross Validation
crossval_lsvc = CrossValidator(estimator = pipeline_lsvc, estimatorParamMaps = paramgrid_lsvc, \
                                evaluator = evaluator_lsvc, numFolds = 3, seed=10)
```

In [37]: *# Training and evaluation*

```
cvModel_lsVC = crossval_lsVC.fit(training)
evaluator_lsVC.evaluate(cvModel_lsVC.transform(test))
```

▶ Spark Job Progress

```
0.9820872274143302
```

In [38]: *# Evaluator*

```
evaluator_lsVC = MulticlassClassificationEvaluator(metricName = "f1")
```

In [39]: *# Cross Validation*

```
crossval_lsVC = CrossValidator(estimator = pipeline_lsVC, estimatorParamMaps = paramgrid_lsVC, \
                                evaluator = evaluator_lsVC, numFolds = 3, seed=10)
```

In [40]: *# Training and evaluation*

```
cvModel_lsVC = crossval_lsVC.fit(training)
evaluator_lsVC.evaluate(cvModel_lsVC.transform(test))
```

▶ Spark Job Progress

```
0.9819830449526602
```

4.4. Random Forest Classifier

In [41]: *# Model Creation*

```
rf = RandomForestClassifier()
```

In [42]: *# Creating Pipeline*

```
pipeline_rf = Pipeline(stages = [assembler, scaler, rf])
```

```
In [43]: # Parameter
paramgrid_rf = ParamGridBuilder().build()
```

```
In [44]: # Evaluator
evaluator_rf = MulticlassClassificationEvaluator(metricName = "accuracy")
```

```
In [45]: # Cross Validation
crossval_rf = CrossValidator(estimator = pipeline_rf, estimatorParamMaps = paramgrid_rf, \
                           evaluator = evaluator_rf, numFolds = 3, seed=10)
```

```
In [46]: # Training and evaluation
cvModel_rf = crossval_rf.fit(training)
evaluator_rf.evaluate(cvModel_rf.transform(test))
```

```
Exception in thread cell_monitor-46:
Traceback (most recent call last):
  File "/mnt/notebook-env/lib/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/mnt/notebook-env/lib/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/mnt/notebook-env/lib/python3.7/site-packages/awseditorssparkmonitoringwidget-1.0-py3.7.egg/awseditors
sparkmonitoringwidget/cellmonitor.py", line 178, in cell_monitor
    job_binned_stages[job_id][stage_id] = all_stages[stage_id]
KeyError: 2692
```

0.9774143302180686

```
In [47]: # Evaluator
evaluator_rf = MulticlassClassificationEvaluator(metricName = "f1")
```

```
In [48]: # Cross Validation
crossval_rf = CrossValidator(estimator = pipeline_rf, estimatorParamMaps = paramgrid_rf, \
                           evaluator = evaluator_rf, numFolds = 3, seed=10)
```

In [49]: *# Training and evaluation*

```
cvModel_rf = crossval_rf.fit(training)
evaluator_rf.evaluate(cvModel_rf.transform(test))
```

```
Exception in thread cell_monitor-49:
Traceback (most recent call last):
  File "/mnt/notebook-env/lib/python3.7/threading.py", line 926, in _bootstrap_inner
    self.run()
  File "/mnt/notebook-env/lib/python3.7/threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "/mnt/notebook-env/lib/python3.7/site-packages/awseditorssparkmonitoringwidget-1.0-py3.7.egg/awseditors
sparkmonitoringwidget/cellmonitor.py", line 178, in cell_monitor
    job_binned_stages[job_id][stage_id] = all_stages[stage_id]
KeyError: 2780
```

0.9772829697229194

4.5. Gradient Boosted Tree Classifier

In [50]: *# Model Creation*

```
gbt = GBTCClassifier()
```

In [51]: *# Creating Pipeline*

```
pipeline_gbt = Pipeline(stages = [assembler, scaler, gbt])
```

In [52]: *# Parameter*

```
paramgrid_gbt = ParamGridBuilder().build()
```

In [53]: *# Evaluator*

```
evaluator_gbt = MulticlassClassificationEvaluator(metricName = "accuracy")
```

```
In [54]: # Cross Validation
crossval_gbt = CrossValidator(estimator = pipeline_gbt, estimatorParamMaps = paramgrid_gbt, \
                                evaluator = evaluator_gbt, numFolds = 3, seed=10)
```

```
In [55]: # Training and evaluation
cvModel_gbt = crossval_gbt.fit(training)
evaluator_gbt.evaluate(cvModel_gbt.transform(test))
```

▶ Spark Job Progress

0.9883177570093458

```
In [56]: # Evaluator
evaluator_gbt = MulticlassClassificationEvaluator(metricName = "f1")
```

```
In [57]: # Cross Validation
crossval_gbt = CrossValidator(estimator = pipeline_gbt, estimatorParamMaps = paramgrid_gbt, \
                                evaluator = evaluator_gbt, numFolds = 3, seed=10)
```

```
In [58]: # Training and evaluation
cvModel_gbt = crossval_gbt.fit(training)
evaluator_gbt.evaluate(cvModel_gbt.transform(test))
```

▶ Spark Job Progress

0.9882996075449674

```
In [59]: # spark.sparkContext.stop()
spark.stop()
```

▶ Spark Job Progress

Progress:

An error was encountered:

Invalid status code '400' from <http://localhost:8998/sessions/0/statements/59> (<http://localhost:8998/sessions/0/statements/59>) with error payload: {"msg": "requirement failed: Session isn't active."}

4.4. Consolidado dos resultados

	Standard Scaler	
	Accuracy	F1
Logistic Regression	0,9800	0,9799
Naive Bayes	0,7967	0,7267
Linear Support Vector Machine	0,9837	0,9836
Random Forest Classifier	0,9807	0,9806
Gradient Boosted Tree Classifier	0,9889	0,9889

Tempo de execução	Servers		Dif.
	01 node (*)	05 nodes (*)	
Preparation	01:25,2	00:40,9	00:44,3
Logistic Regression	00:45,3	00:37,2	00:08,1
Naive Bayes	00:28,0	00:17,1	00:10,9
Linear Support Vector Machine	02:32,2	02:13,4	00:18,8
Random Forest Classifier	00:38,7	00:35,3	00:03,4
Gradient Boosted Tree Classifier	01:42,8	01:32,7	00:10,1

(*) node = m5.xlarge (4 vCore, 16 GiB memory)

5. Referência

Página do Professor Paulo Cortez

<http://www3.dsi.uminho.pt/pcortez/Home.html>

Página de download da base de dados (dataset winequality.zip)

<http://www3.dsi.uminho.pt/pcortez/wine/>

Spark Apache

<https://spark.apache.org/>

Apache Spark

<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-spark.html>

Create a cluster with Spark

<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-spark-launch.html>

Apache Hive

<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-hive.html>

Apache Livy

<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-livy.html>

Amazon EMR Notebook based on Jupyter Notebook

<https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-jupyter-emr-managed-notebooks.html>

Spark 3.2.0 - ML Pipelines

<https://spark.apache.org/docs/latest/ml-guide.html>

Machine Learning with PySpark and Amazon EMR

<https://towardsdatascience.com/machine-learning-with-pyspark-and-amazon-emr-3149dbc847ae>

3. Apresentação

- Cada grupo terá no máximo 15 minutos para apresentar o trabalho.
 - Todos os membros do grupo deverão estar presentes na data de sua apresentação.
 - Todos os grupos deverão estar presentes nos dias de apresentação.
 - A banca de avaliação, formada pelos professores das disciplinas, poderá arguir o grupo ou diretamente qualquer membro do grupo.
-

- A banca é soberana e responsável por resolver os casos previstos nestas orientações e definir os casos omissos.
- As datas das apresentações serão definidas durante o semestre juntamente com professores e alunos de todas as disciplinas.

4. Entregáveis

O que será entregue:

- Relatório (pode ser o notebook ou um github bem detalhado e organizado)
- Apresentação (slides)
- Todos os demais artefatos produzidos (*scripts, architecture workflows, jobs*)

5. Critérios de Avaliação

Por ser um projeto interdisciplinar, a banca definirá, em comum acordo, a nota obedecendo-se a proporção de 40% referente à nota geral do projeto, valorada para cada aluno (20% apresentação oral, 10% slides e 10% arguição) e 60% a cargo de cada professor em sua disciplina.