Noeud IoT Pour la maintenance predictive

Réaliser par :
Rabii Bouhelal
Mohamed Elhedi Ben Yedder

Encadrer par:
Mr. Khaled Grati
Mme. Fatma Rouissi

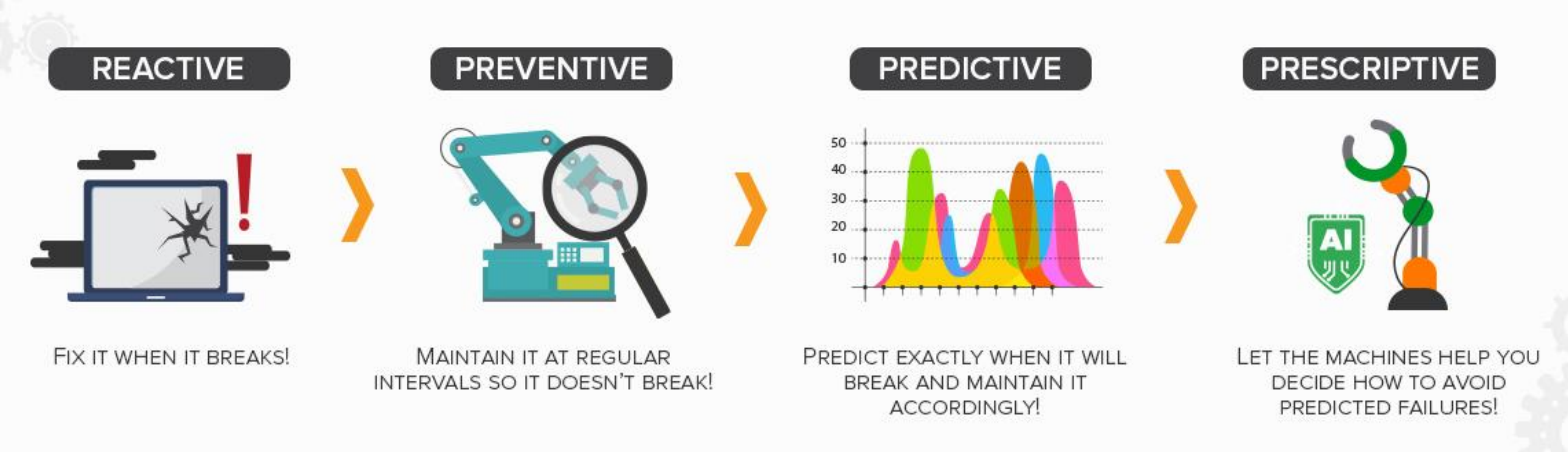
Plan

1. Scenario
2. Realisation
3. Résultats

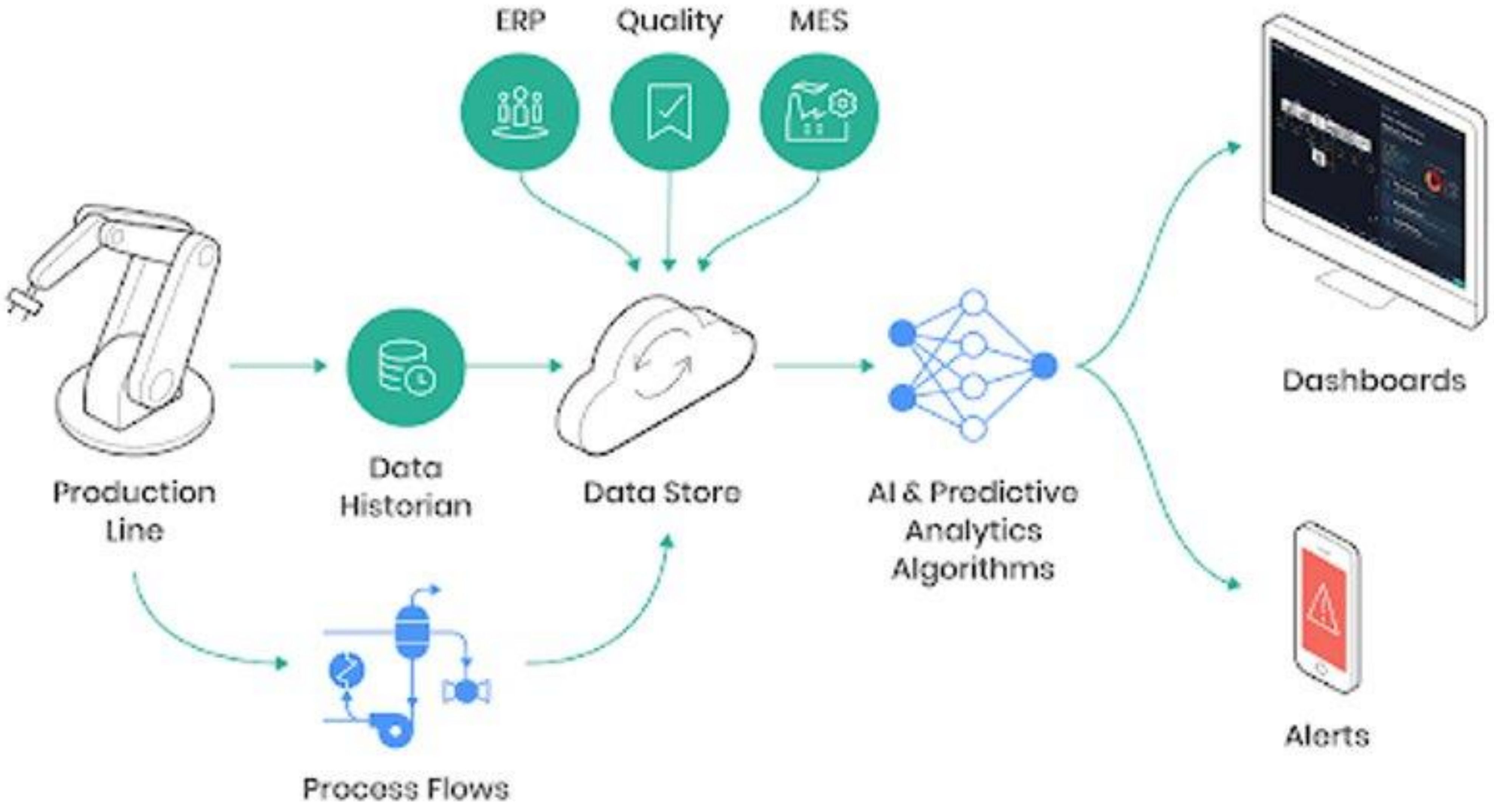
Scenario

1. Problématique
2. Architecture general
3. Spécifications

Problématique

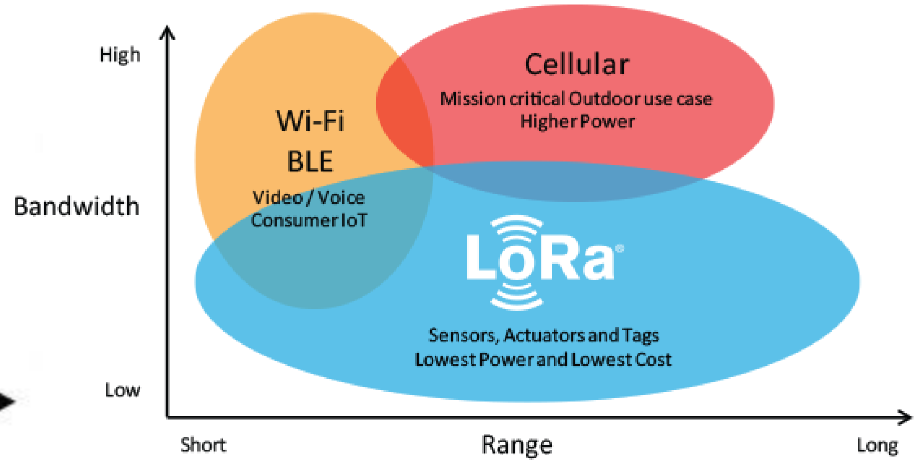


Architecture général



Spécifications

Power	Lower cost and greater power efficiency than other wireless networks and supports a greater number of connected devices over a larger area.
Portability	Devices (sensors, switches) are autonomous—battery powered, wireless and completely cable free.
Range	Range from 1 to 30 km in different environment conditions.
Autonomy	High autonomy of smart devices, with a lifetime up to 10 years.
Data	Data transmitted with low throughput—packet sizes from 10 to 1,000 bytes at uplink speeds up to 200 Kbps.
Cost	Radio modules and chipsets are relatively inexpensive.
Latency	Low latency (although not a key parameter in most IoT applications).
Access points	Fewer access points (base stations, gateways) than other wide-area technologies (ex. cellular) required to cover a wide area such as a city.
Penetration	Good penetration of structures and walls; able to be used underground and inside buildings.
Resistance to exterior Factors	Many devices are environmentally hardened, so can be used year-round in Canadian climates.



Key Factors	
Power	Low consumption
Range	10 -> 15 km
Data rate	10 bps to 10 kbps
Message Size	-----------
Mobility	Low speed Mobility
Traffic pattern	Radom
Data rate	------------
Device Density	Low
Bandwidth	50-100 Hz
Latency tolerance	Minutes to hours

Réalisation

Cliquez pour ajouter du texte

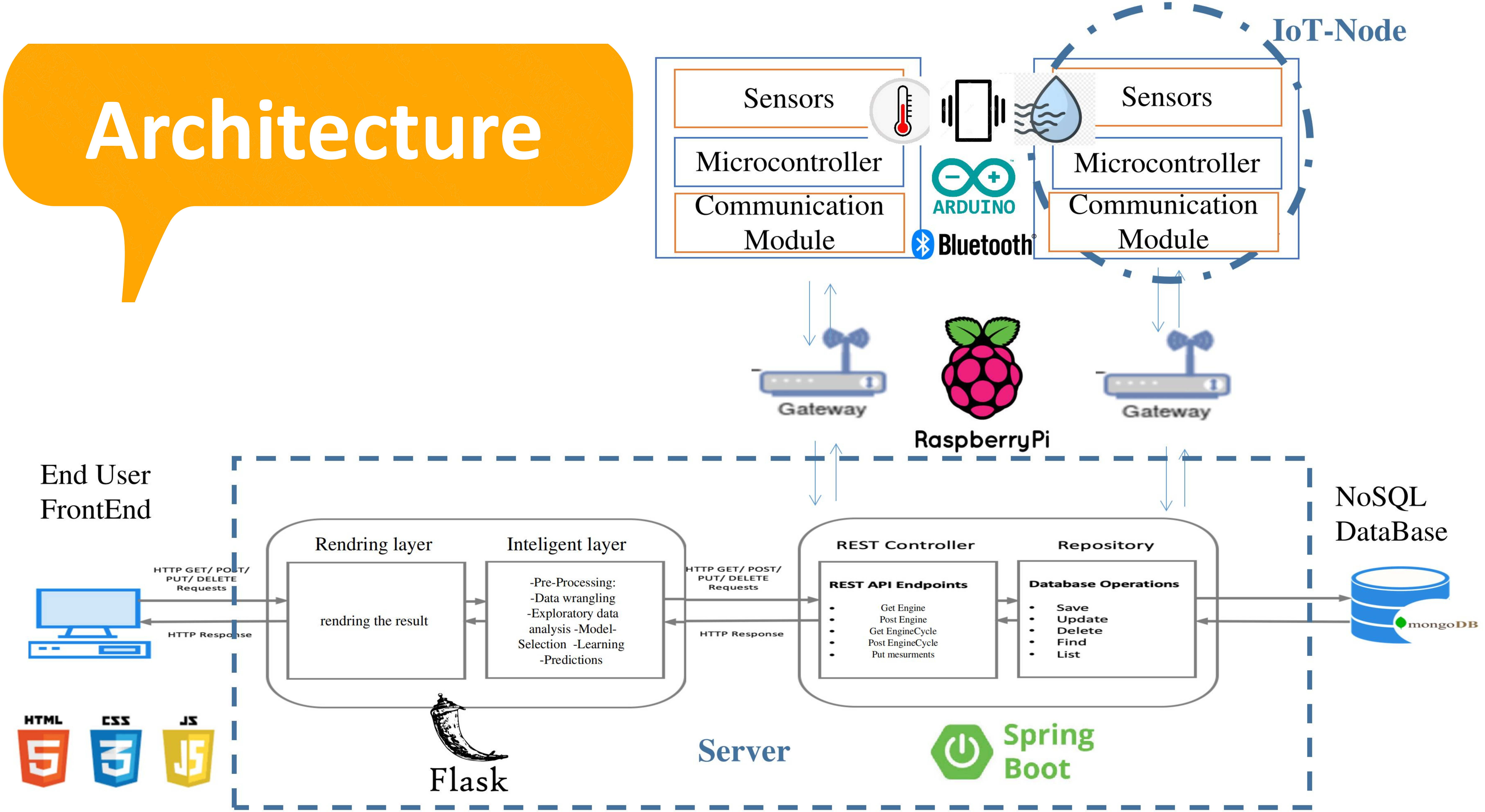
1 Architecture

2 Noeud

3 Gateway

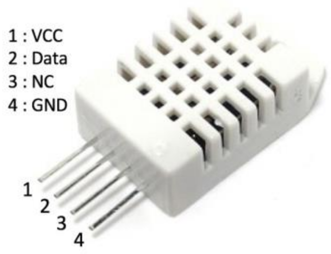
4 Serveur et stockage

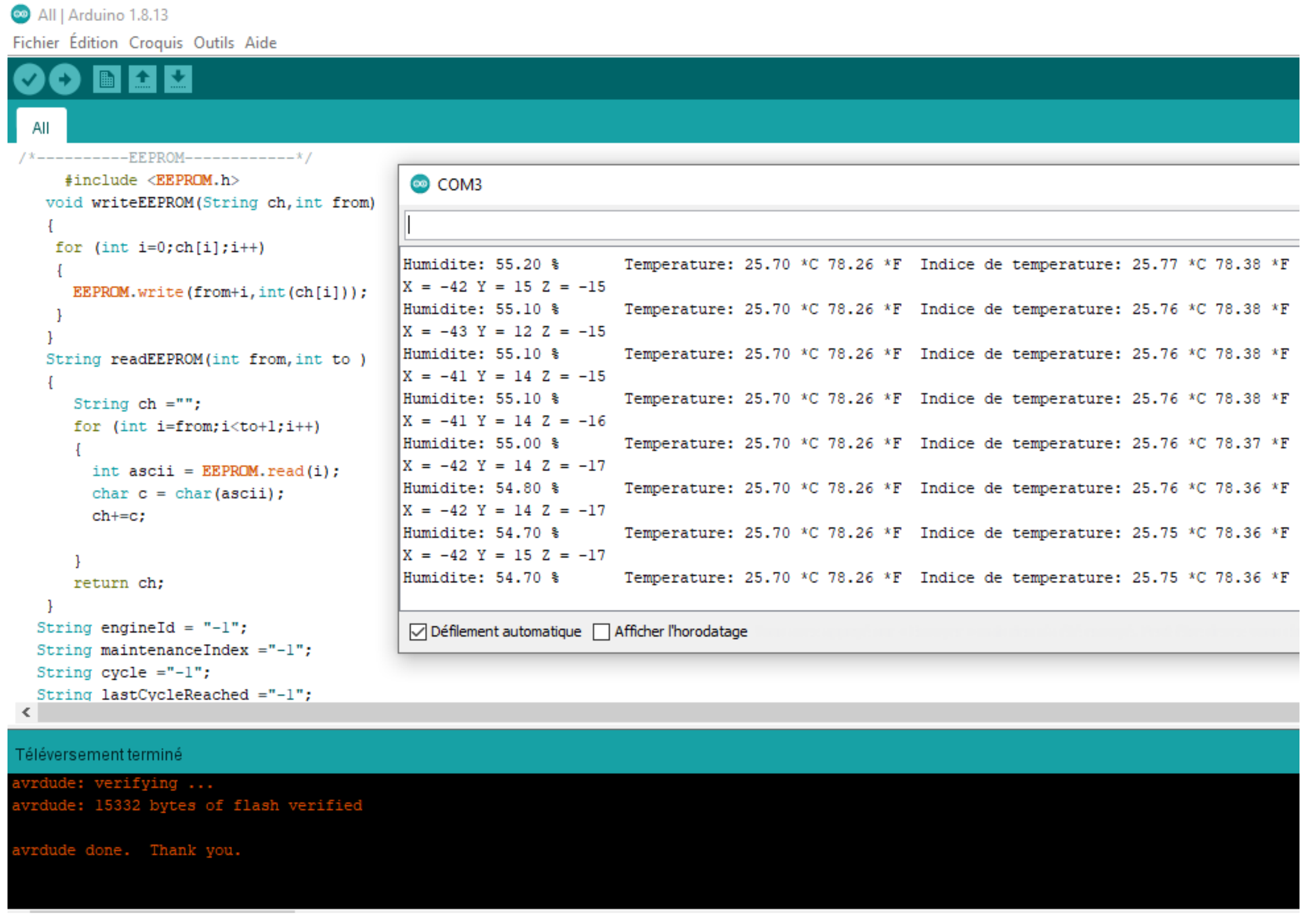
Architecture



Noeud -IoT

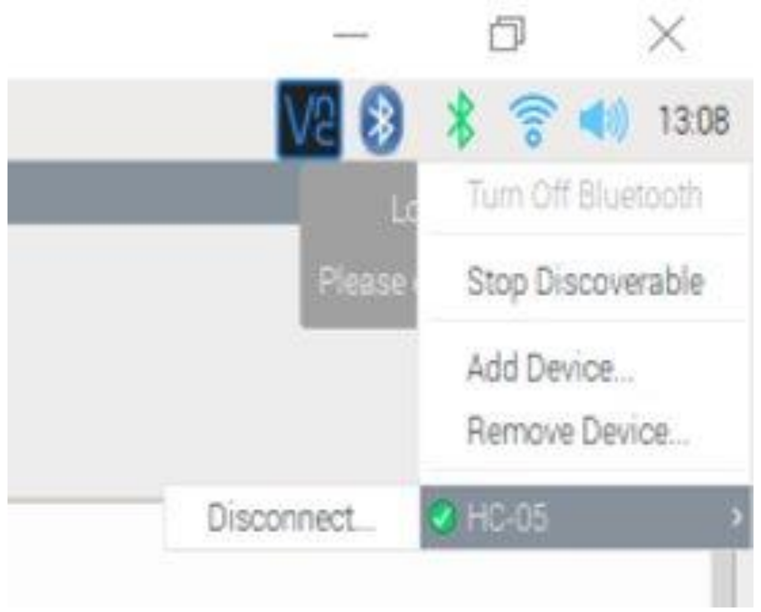
1. Capteurs
2. Microcontroleur
3. Module de communication





GateWay


192.168.137.192 (raspberrypi) - VNC Viewer
Thonny - /home/pi/...
VNC Server - File Tran...
*Nouveau document ...
New
Load
Run
Debug
Stop
Zoom
Quit
BlueTooth.py
ser = serial.Serial(
port='/dev/rfcomm3',
baudrate=9600,
parity=serial.PARITY_ODD,
stopbits=serial.STOPBITS_TWO,
bytesize=serial.SEVENBITS
)
ser.isOpen()
print('Enter your message below.\r\nInsert "exit" to leave the application.')
delay = 10000
while 1 :
user_input = input("Server message >> ")
if user_input == 'exit':
ser.close()
myobj={'engineCycle': {"id": {
"engine": {
"id": 10,
"description": "motor10",
"maintenanceIndex": 0
},
"cycle": 0,
"maintenanceIndex": 0
}},"measurements":L}
print(rqs.post("http://192.168.131.1:8080/api/engineCycle/addMeasurement",data=
exit()
Shell
Device Response >> -39.00,16.00,-16.00,25.80,53.80,25.84
Server message >> 111111111111
Device Response >> -41.00,16.00,-18.00,25.70,53.70,25.73
Server message >> 1111111111111111
Device Response >> -40.00,15.00,-16.00,25.70,53.70,25.73
Server message >> 1111111111111
Device Response >> -40.00,16.00,-18.00,25.70,53.60,25.72
Server message >> 11111111111111111111
Device Response >> -40.00,16.00,-17.00,25.70,53.70,25.73
Server message >>


13:08
Turn Off Bluetooth
Stop Discoverable
Add Device...
Remove Device...
Disconnect...
HC-05

Serveur & stockage

1. Base de données
2. Couche de données
3. Couche intelligente

1

Base de données

1 Database

mongoDB

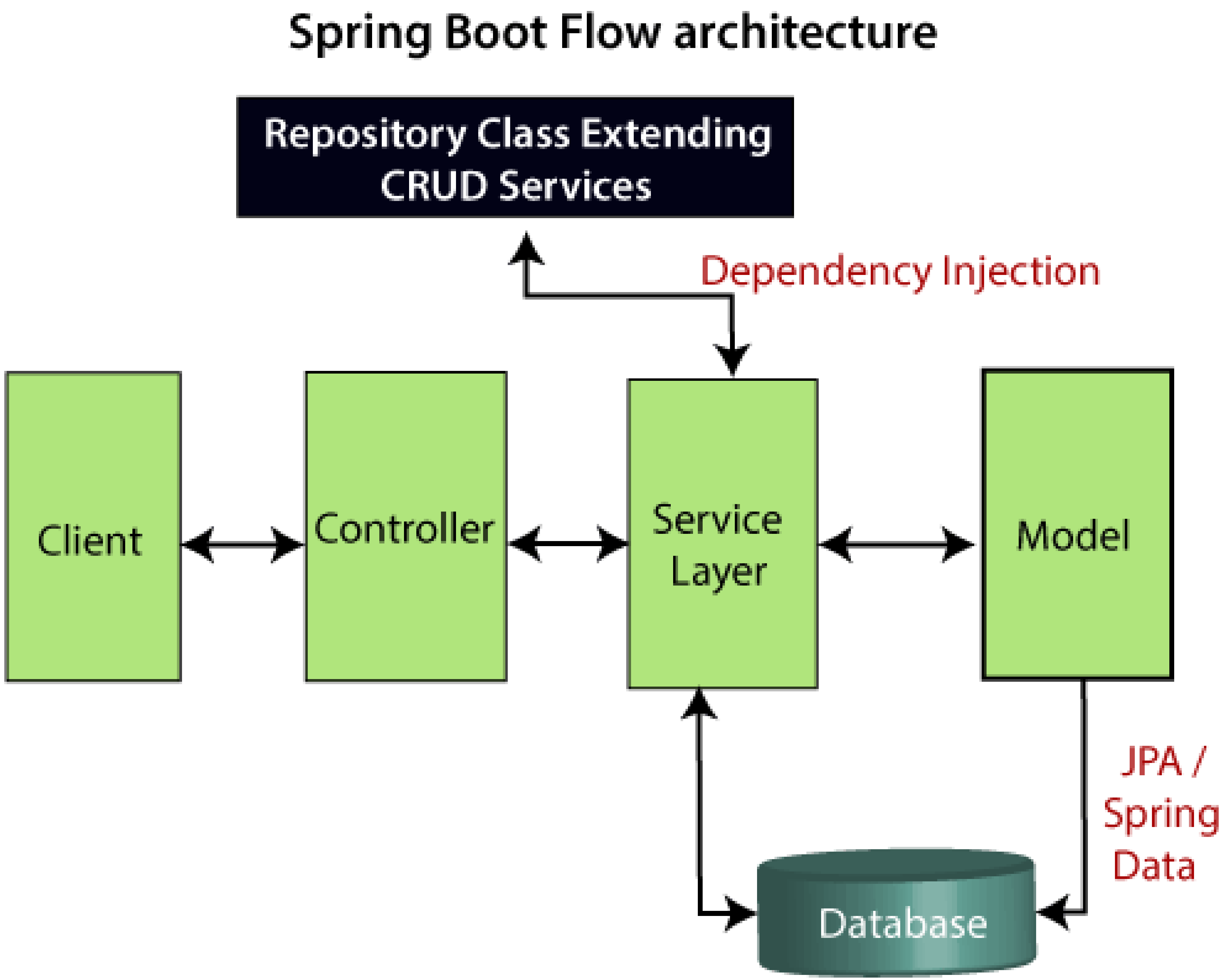
Engine	100	88.4 B	8.6 KB	1	16.0 KB
EngineCycle	5	5.5 KB	27.6 KB	1	36.0 KB

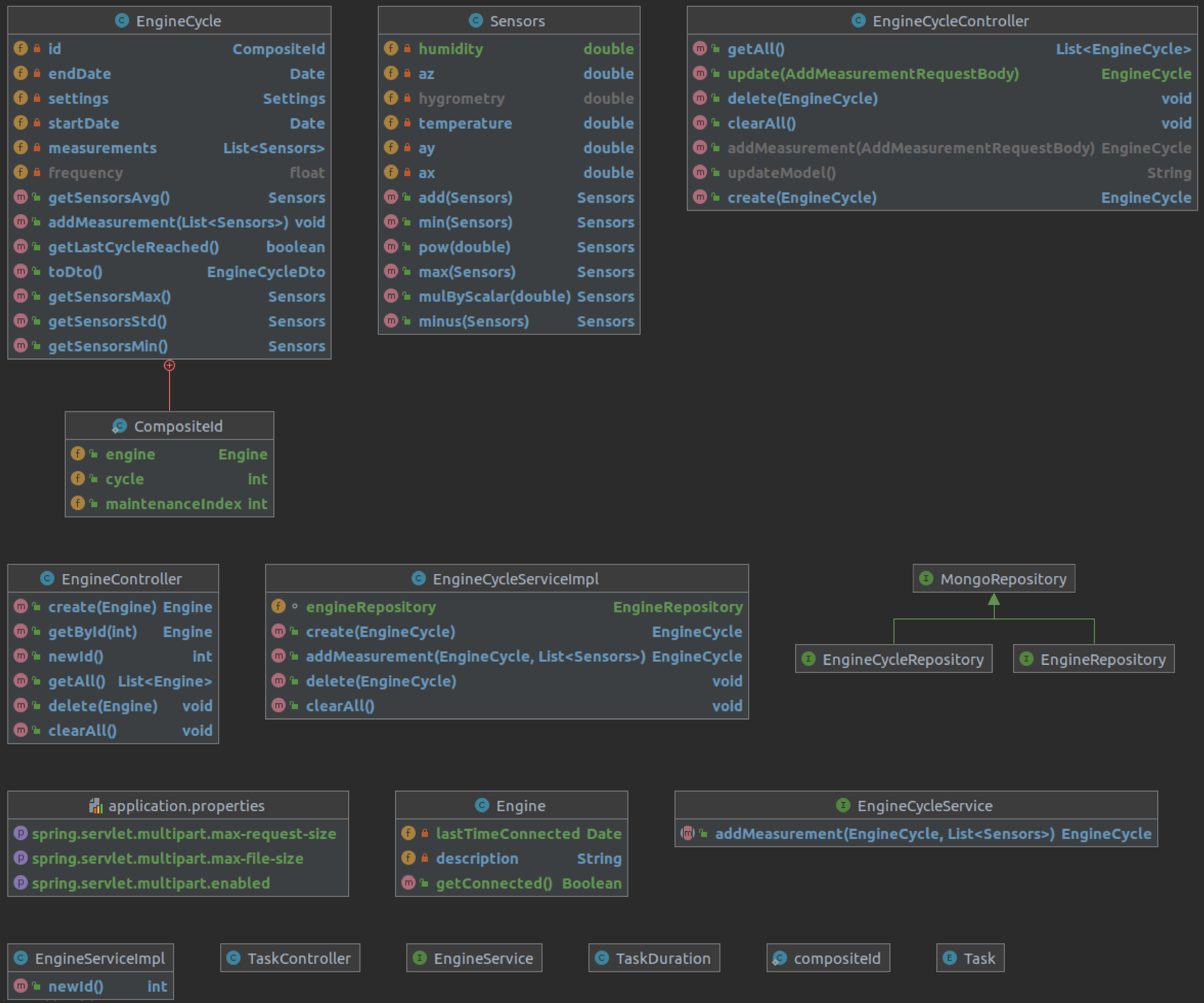
object {4}
_id : 1
description : vehicula. Pellentesque tincidunt tempus risus. Donec egestas. Duis ac arcu.
maintenanceIndex : 0
lastTimeConnected {1}
$date : 2021-06-07T23:00:00.000Z

object {4}
_id {3}
engine {2}
$ref : Engine
$id : 1
cycle : 0
maintenanceIndex : 0
measurementFrequency : 0
measurements [26]
0 {5}
ax : 82.30215
ay : 43.880162
az : -66.725855
temperature : 37.5591
humidity : 69.1895
1 {5}
2 {5}
3 {5}
4 {5}
5 {5}
6 {5}

2

Couche de données


Spring Boot Flow architecture
Repository Class Extending CRUD Services
Dependency Injection
Client
Controller
Service Layer
Model
JPA / Spring Data
Database


EngineCycle
id CompositeId
endDate Date
settings Settings
startDate Date
measurements List<Sensors>
frequency float
getSensorsAvg() Sensors
addMeasurement(List<Sensors>) void
getLastCycleReached() boolean
toDto() EngineCycleDto
getSensorsMax() Sensors
getSensorsStd() Sensors
getSensorsMin() Sensors
Sensors
humidity double
az double
hygrometry double
temperature double
ay double
ax double
add(Sensors) Sensors
min(Sensors) Sensors
pow(double) Sensors
max(Sensors) Sensors
mulByScalar(double) Sensors
minus(Sensors) Sensors
EngineCycleController
getAll() List<EngineCycle>
update(AddMeasurementRequestBody) EngineCycle
delete(EngineCycle) void
clearAll() void
addMeasurement(AddMeasurementRequestBody) EngineCycle
updateModel() String
create(EngineCycle) EngineCycle
CompositeId
engine Engine
cycle int
maintenanceIndex int
EngineController
create(Engine) Engine
getById(int) Engine
newId() int
getAll() List<Engine>
delete(Engine) void
clearAll() void
EngineCycleServiceImpl
engineRepository EngineRepository
create(EngineCycle) EngineCycle
addMeasurement(EngineCycle, List<Sensors>) EngineCycle
delete(EngineCycle) void
clearAll() void
MongoRepository
EngineCycleRepository
EngineRepository
application.properties
spring.servlet.multipart.max-request-size
spring.servlet.multipart.max-file-size
spring.servlet.multipart.enabled
Engine
lastTimeConnected Date
description String
getConnected() Boolean
EngineCycleService
addMeasurement(EngineCycle, List<Sensors>) EngineCycle
EngineServiceImpl
newId() int
TaskController
EngineService
TaskDuration
compositeId
Task

2 Couche de données

id			lastCycleReached	Sensors_Avg			Sensors_Std			Sensors_Min			Sensors_Max		
Engine	cycle	maintenanceIndex		1	..	N	1	..	N	1	...	N	1		N
......	.....	......	.....	......	.... .	... ...	......	.....	......	......	.....	......	......	.... .	......
......	.....	......	.....	......	.... .	... ...	......	.....	......	......	.....	......	......	.... .	......
......	.....	......	.....	......	.... .	... ...	......	.....	......	......	.....	......	......	.... .	......

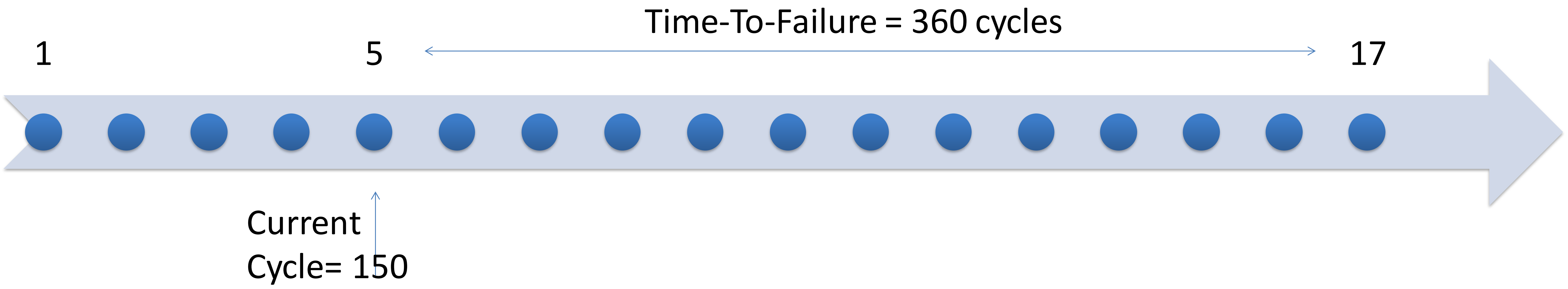
3

Couche intelligente

Flask

L'approche

1. Utilisation d'un algorithme regression pour prevoire TTF (Time-To-Failure)
2. Utilisation d'un algorithme de classificatio pour prevoire si le moteur va tomber en panne dans cette période
3. Utilisation d'un algorithme de classification multi-classe pour prévoire la période dans laquelle va tomber en panne



Regression	Binary Classification	Multiclass Classification
TTF (Time-To-Failure)	if the engine will fail in this period	the period an engine will fail
360 cycles	0	0

Pre-traitement de données

1. Extraction de nouveaux descripteurs (feature Extraction)
2. Calcul des labels

1

Extraction de nouveaux descripteurs (feature Extraction)

- Definition d'une fenetre (Periode)
- Ajout de nouveaux descripteurs (moyenne glissante et ecart type glissant sur la periode definie ....)
- => à chaque descripteur est associé:
  - ...._Avg
  - ...._Std
  - ...._Max
  - ...._Min
- Objectif : faire apparaitre de nouveaux descripteurs significatifs en fonction du temps

	Date	Close*	Rolling Close Average
10	Feb 01, 2019	62.44	NaN
9	Feb 04, 2019	62.58	62.510
8	Feb 05, 2019	64.27	63.425
7	Feb 06, 2019	63.44	63.855
6	Feb 07, 2019	61.50	62.470
5	Feb 08, 2019	61.16	61.330
4	Feb 11, 2019	62.57	61.865
3	Feb 12, 2019	62.36	62.465
2	Feb 13, 2019	61.63	61.995
1	Feb 14, 2019	60.75	61.190
0	Feb 15, 2019	61.58	61.165

```
def add_features(df_in, rolling_win_size,columns_to_treat):

    """Add rolling average and rolling standard deviation for sensors readings using
fixed rolling window size.

    Args:
            df_in (dataframe)     : The input dataframe to be proccessed (training or
test)
            rolling_win_size (int): The window size, number of cycles for applying
the rolling function
            columns_to_treat      : the list of features to take in consideration and
treat them
    Reurns:
            dataframe: contains the input dataframe with additional rolling mean ,std,
min and max for each sensor

    """
```

1 Calcul des labels

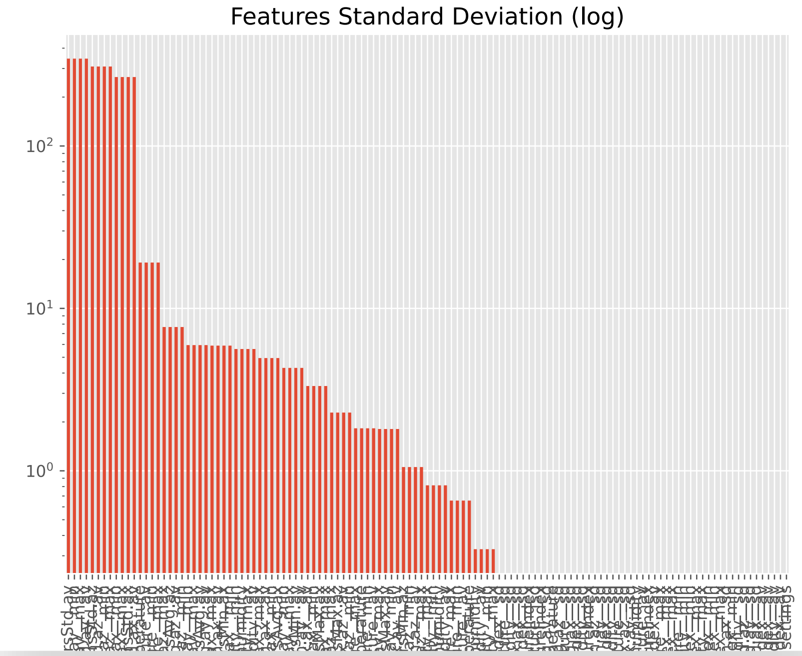
- **TTF :** Time-To-Failure = lastCycle – CurentCycle
- **BNC** : Binary-Class-Classification
  label = 1 si TTF <= Periode sinon 0
- **MCC** : Multi-Class-Classification
  label = 2 si TTF <= Periode/2 sinon 1 si TTF <= Periode sinon 0

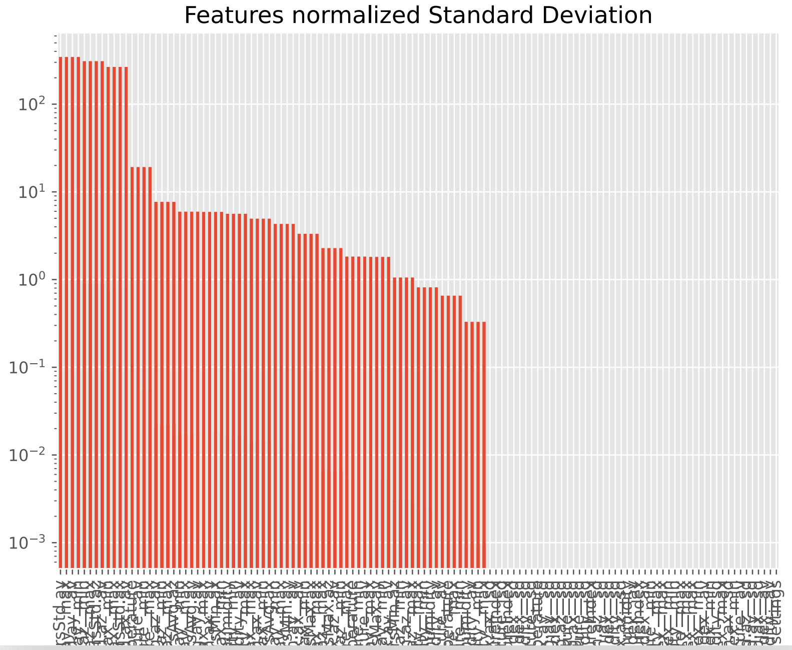
```
def find_labels(df_in, period):

    """Add regression and classification labels to the training data.

        Regression label: labels.ttf (time-to-failure) = each cycle# for an engine
subtracted from the last cycle# of the same engine
        Binary classification label: labels.bnc = if ttf is <=  period then 1 else 0
(values = 0,1)
        Multi-class classification label: labels.mcc = 2 if ttf <= 0.5* period , 1
if ttf<= period, else 2

      Args:
          df_in (dataframe): The input training data
          period (int)     : The number of cycles for TTF segmentation. Used to
derive classification labels

      Returns:
          dataframe: The input dataframe with regression and classification labels
added

    """
```

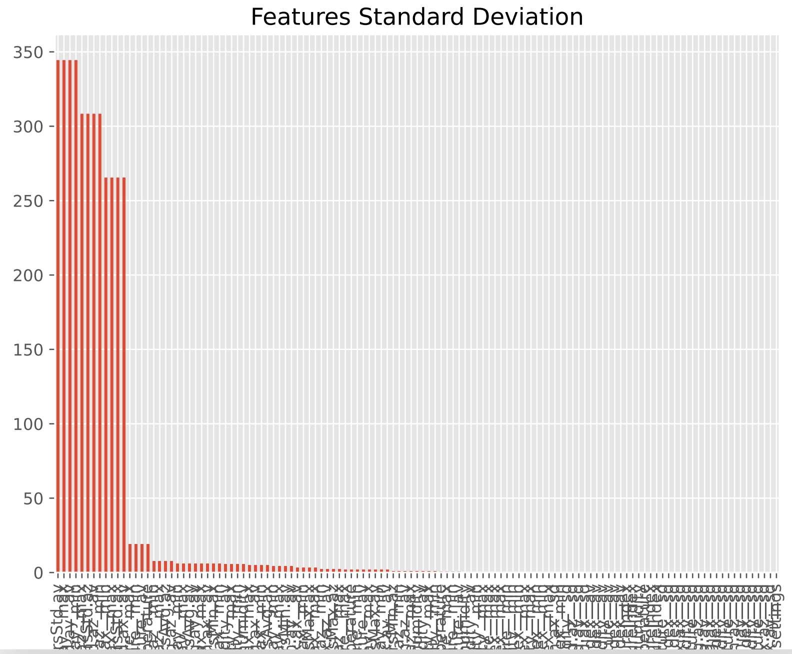
Analyse exploratoire des données (EDA)

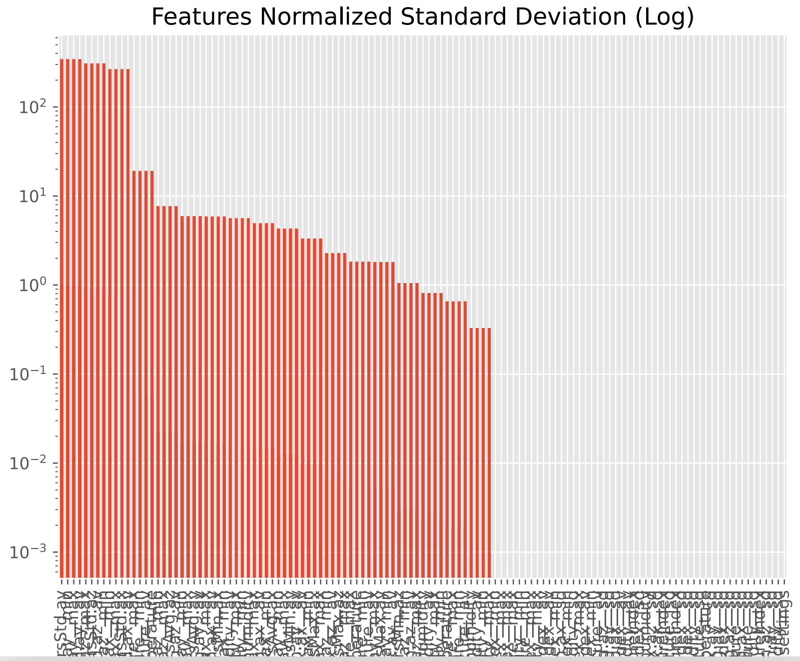
1. La variance
2. Intercorrelation avec le label
2. Intercorrelation entre les descripteurs entre eux

1





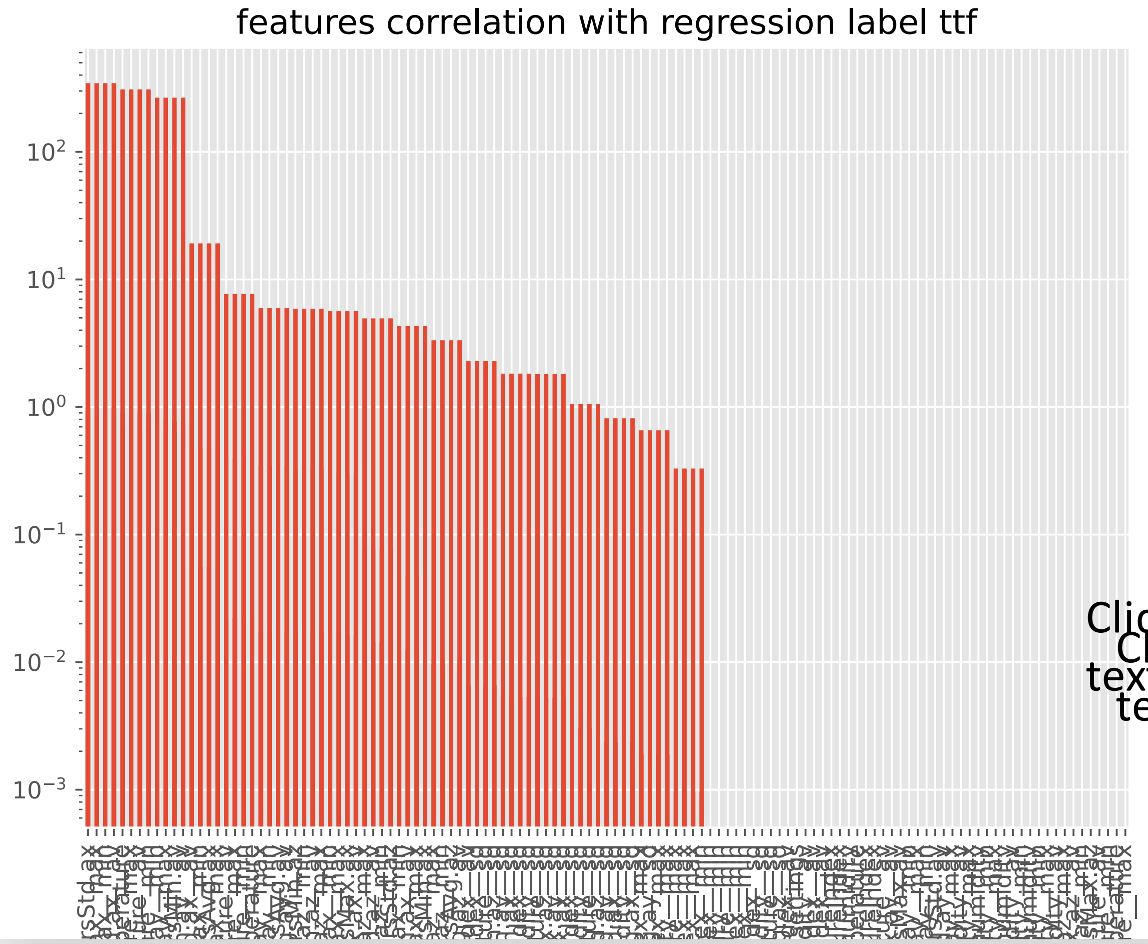




La variance

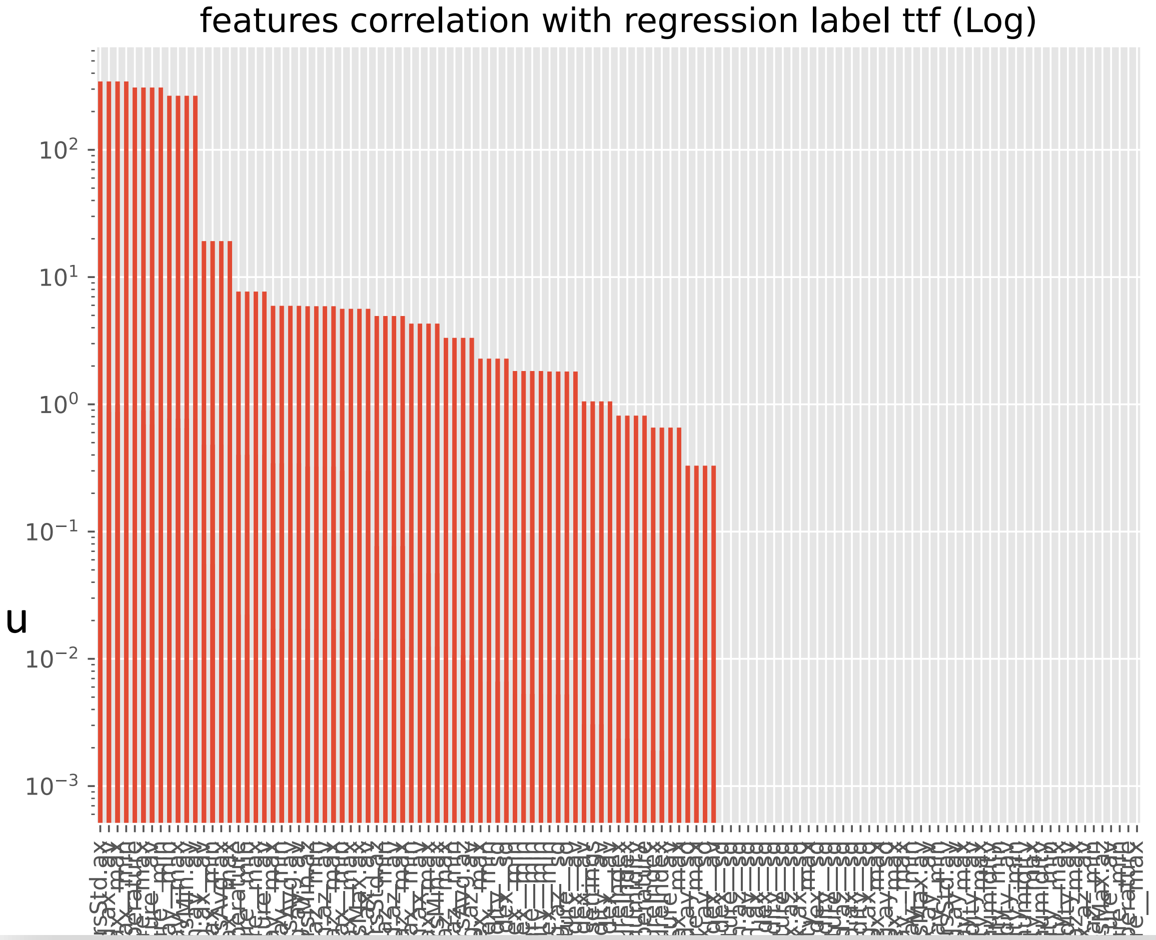
- But : Maximiser la variance

2



Cliquez pour ajouter du texte

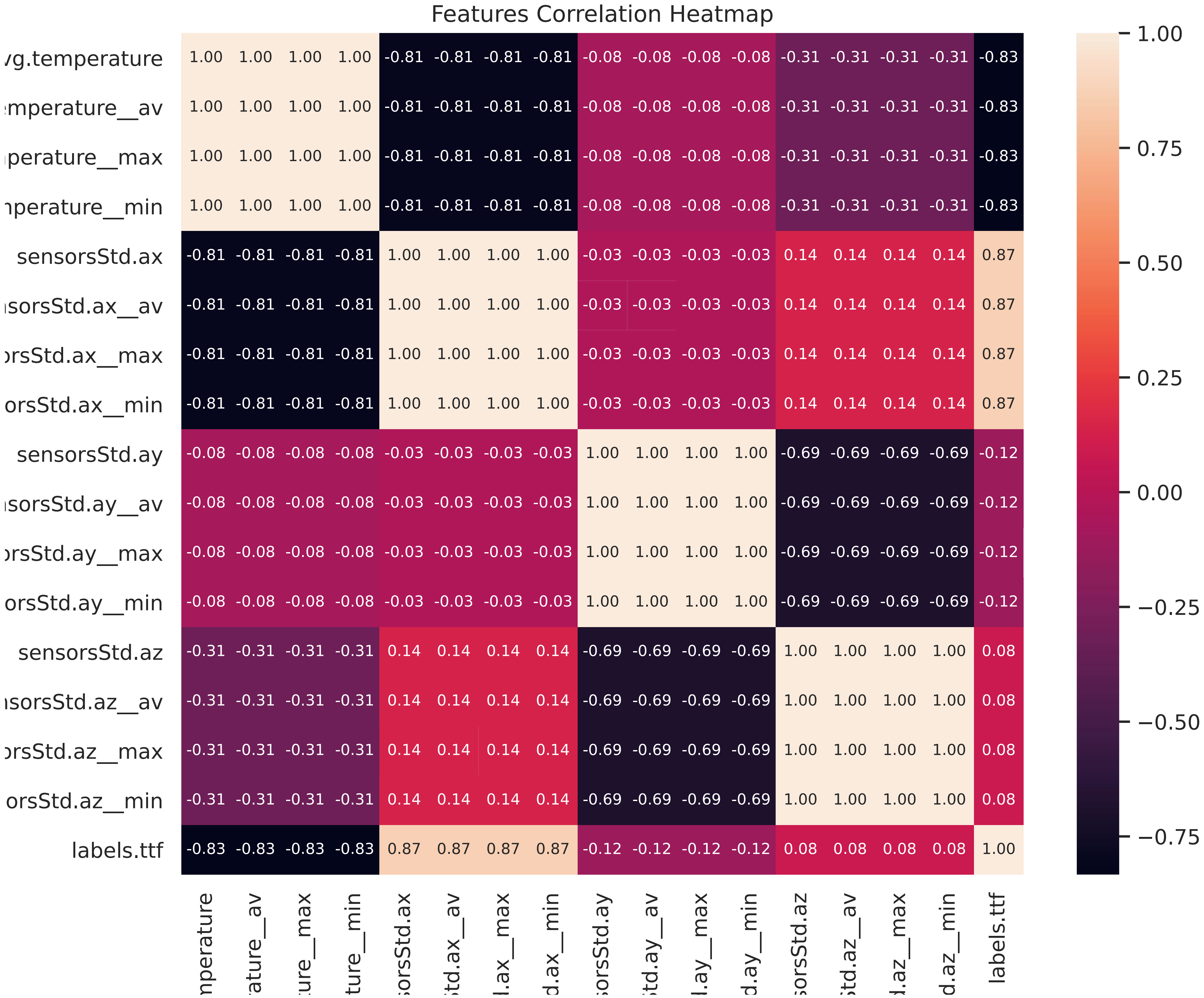
Cliquez pour ajouter du texte



Intercorrelation avec le label

3

Intercorrelation entre les descripteurs (entre eux)



1

Predicting Engine's Time-To-Failure (TTF)

2

Which Engines will fail in the Current Period? (BNC)

3

How could Maintenance be better Planned? (MCC)

Selection des modèles

1

Regression algorithms	Regression metrics
Linear Regression LASSO Regression Ridge Regression Decision Tree Regression Random Forest	R-squared (R^2) Root Mean Squared Error (RMSE) Mean Absolute Error Explained Variance

2

Binary Classification algorithms	Binary Classification metrics
Logistic Regression Decision Trees Support Vector Machines Linear Support Vector Gaussian Naive Random Forests	Precision Recall F1 Score Accuracy

3

Multiclass Classification algorithms	Multiclass Classificationmetrics
Logistic Regression Decision Trees Support Vector Machines Linear Support Vector Gaussian Naive Random Forests	Precision Recall F1 Score Accuracy

TTF

	index	LinearRegression	DecisionTreeClassifier	RandomForestClassifier	SVC	LinearSVC	GaussianNB
0	Accuracy	0.500000	0.5	0.0	0.5	0.500000	0.5
1	Precision	0.500000	0.0	0.0	0.0	0.500000	0.0
2	Recall	1.000000	0.0	0.0	0.0	1.000000	0.0
3	F1 Score	0.666667	0.0	0.0	0.0	0.666667	0.0

BNC

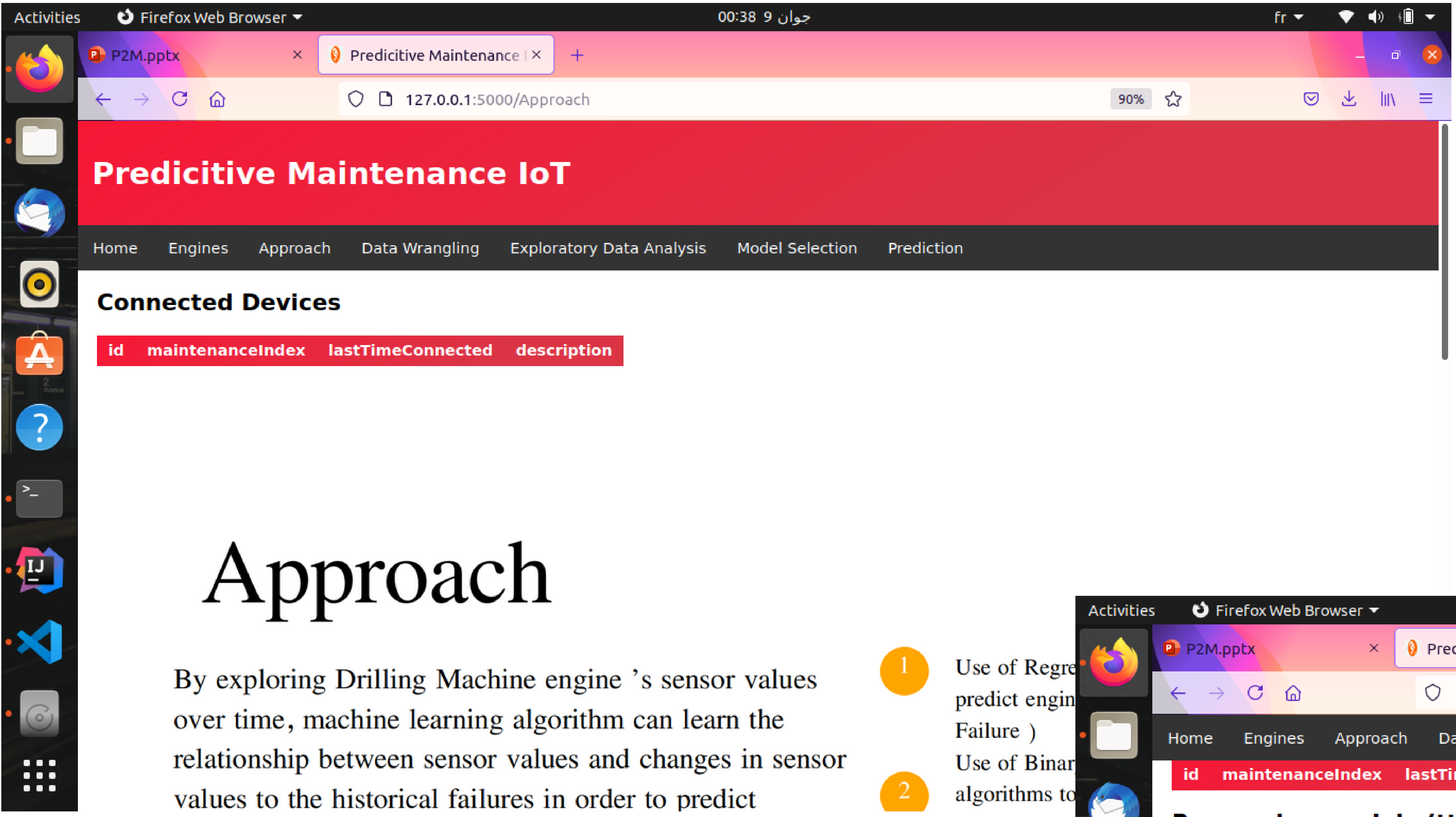
	index	LogisticRegression	DecisionTreeClassifier	RandomForestClassifier	SVC	LinearSVC	GaussianNB
0	Accuracy	0.500000	0.5	0.0	0.5	0.500000	0.5
1	Precision	0.500000	0.0	0.0	0.0	0.500000	0.0
2	Recall	1.000000	0.0	0.0	0.0	1.000000	0.0
3	F1 Score	0.666667	0.0	0.0	0.0	0.666667	0.0

MCC

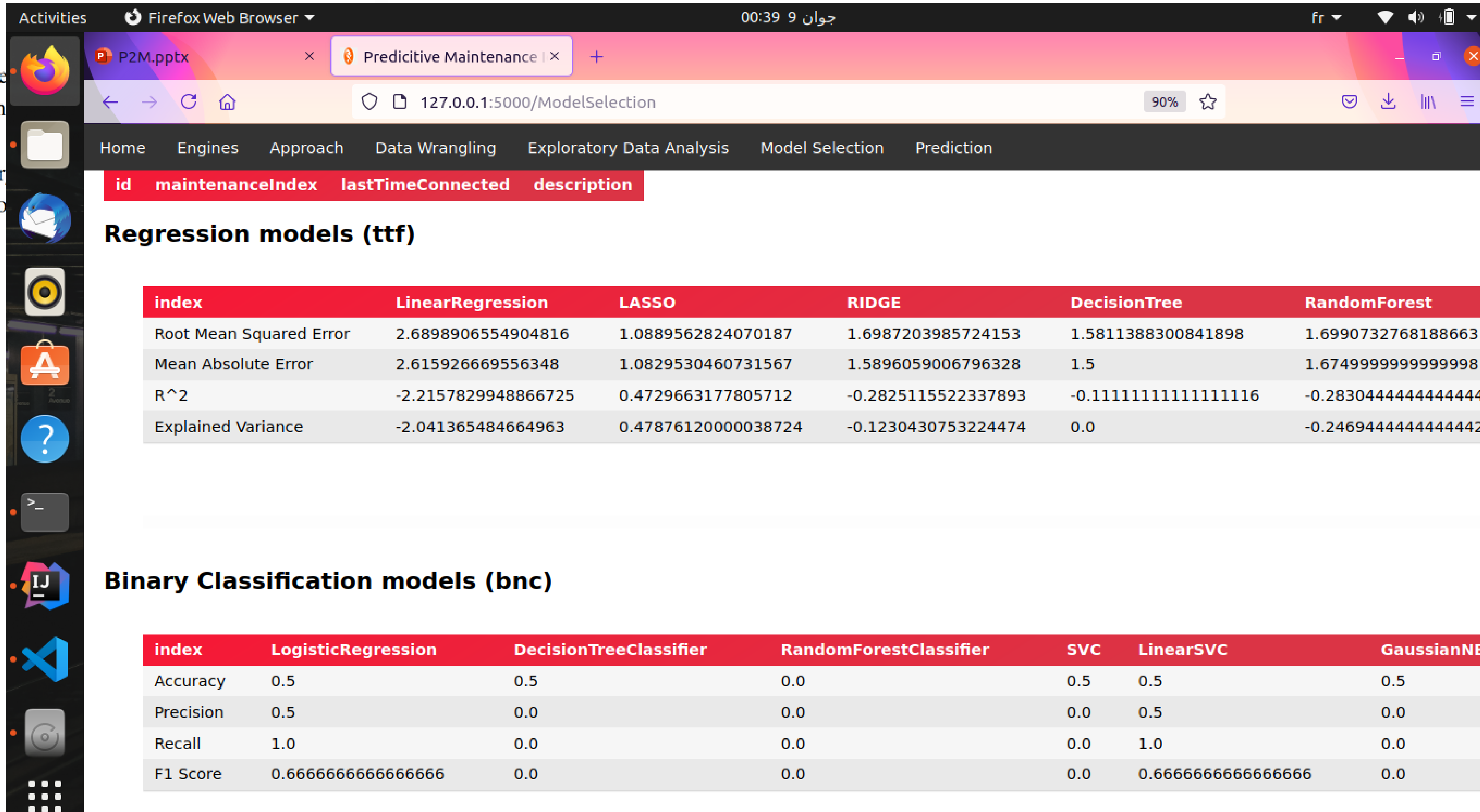
index	LogisticRegression	DecisionTreeClassifier	RandomForestClassifier	SVC	LinearSVC	GaussianNB
Accuracy	0.500000	0.500000	0.0	0.500000	0.500000	0.500000
macro F1	0.333333	0.333333	0.0	0.333333	0.333333	0.333333
micro F1	0.500000	0.500000	0.0	0.500000	0.500000	0.500000
macro Precision	0.250000	0.250000	0.0	0.250000	0.250000	0.250000
micro Precision	0.500000	0.500000	0.0	0.500000	0.500000	0.500000
macro Recall	0.500000	0.500000	0.0	0.500000	0.500000	0.500000
micro Recall	0.500000	0.500000	0.0	0.500000	0.500000	0.500000

Résulats

1. Interface utilisateur (Front End User)
2. Prédiction



1 Interface utilisateur (Front End User)



Regression models (ttf)

index	LinearRegression	LASSO	RIDGE	DecisionTree	RandomForest
Root Mean Squared Error	2.6898906554904816	1.0889562824070187	1.6987203985724153	1.5811388300841898	1.6990732768188663
Mean Absolute Error	2.615926669556348	1.0829530460731567	1.5896059006796328	1.5	1.6749999999999998
R^2	-2.2157829948866725	0.4729663177805712	-0.2825115522337893	-0.11111111111111116	-0.2830444444444444
Explained Variance	-2.041365484664963	0.47876120000038724	-0.1230430753224474	0.0	-0.2469444444444442

Binary Classification models (bnc)

index	LogisticRegression	DecisionTreeClassifier	RandomForestClassifier	SVC	LinearSVC	GaussianNB
Accuracy	0.5	0.5	0.0	0.5	0.5	0.5
Precision	0.5	0.0	0.0	0.0	0.5	0.0
Recall	1.0	0.0	0.0	0.0	1.0	0.0
F1 Score	0.6666666666666666	0.0	0.0	0.0	0.6666666666666666	0.0

2 Prédiction

Results

we used the following algorithms selected after model selection step to predict TTF BNC MCC respectively

LinearRegression **LogisticRegression** **LogisticRegression**

id.engine.id	id.engine.maintenanceIndex	id.engine.lastTimeConnected	id.engine.description	id.maintenanceIndex	labels.ttf	labels.bnc	labels.mcc
1	0	0	vehicula. Pellentesque tincidunt tempus risus. Donec egestas. Duis ac arcu.	0	4.000000000000021	0	0
1	0	0	vehicula. Pellentesque tincidunt tempus risus. Donec egestas. Duis ac arcu.	0	3.0000000000000355	0	0
1	0	0	vehicula. Pellentesque	0	2.000000000000007	0	0

Merci Pour votre attention

bonne journée