INSE

Práctica: Smart Health

Reconocimiento de actividades físicas basado en acelerómetro 3D

Cristina Luna Jiménez,

Fernando Fernández-Martínez,

Jose Manuel Pardo Muñoz,

fernando.fernandezm@upm.es

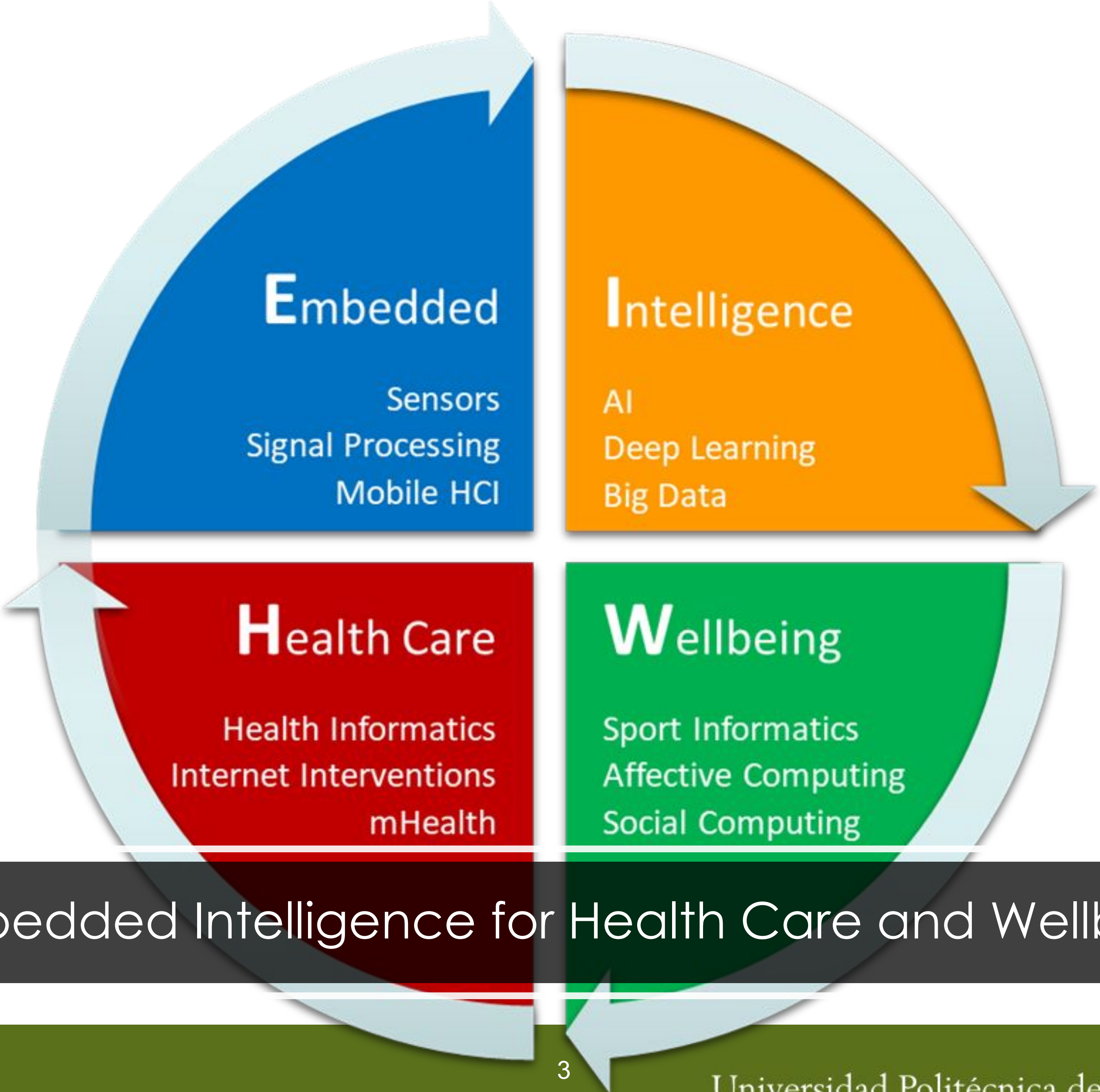
POLITÉCNICA

"Ingeniamos el futuro"

Departamento de Ingeniería Electrónica

Revisión 11, Octubre, 2020

Contents



Embedded Intelligence for Health Care and Wellbeing

a b c

d e f

Sensors embedded in everyday surroundings

a) armband
b) ECG monitoring on a car seat
c) neonatal monitoring
d) ear worn activity recognition
e) Nike+iPod for monitoring running
f) wheelchair motion and paralympic athlete performance

Medical Internet of Things (MIoT)

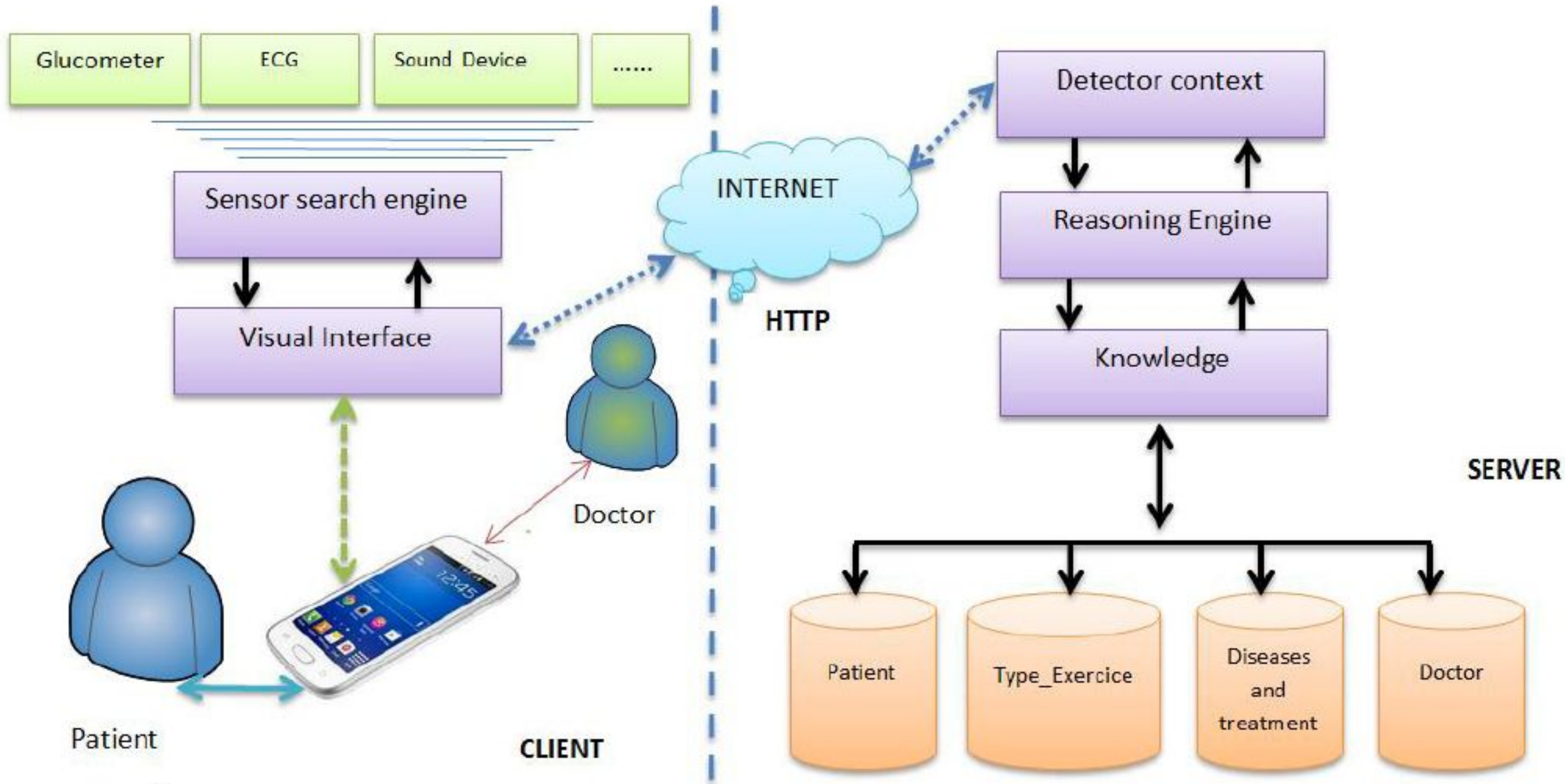
- Integration of medical devices in a network connection
- Network can be managed from the web
- Provide information in real time
- Communication: person to person (P2P) & machine to machine (M2M)
- Allow interaction between health professionals & patients
- MIoT can be seen from 3 paradigms:
  - Internet-oriented middleware
  - Things sensors oriented
  - Knowledge-oriented semantics

Embedded Intelligence in MIoT

- Growth at a high rate exceeding 7%
- Estimated Revenues by 2020 $2.2 trillion
- Healthcare is one of the leading industries
  - Source: 3rd WHO Global Forum on Medical Devices (link)

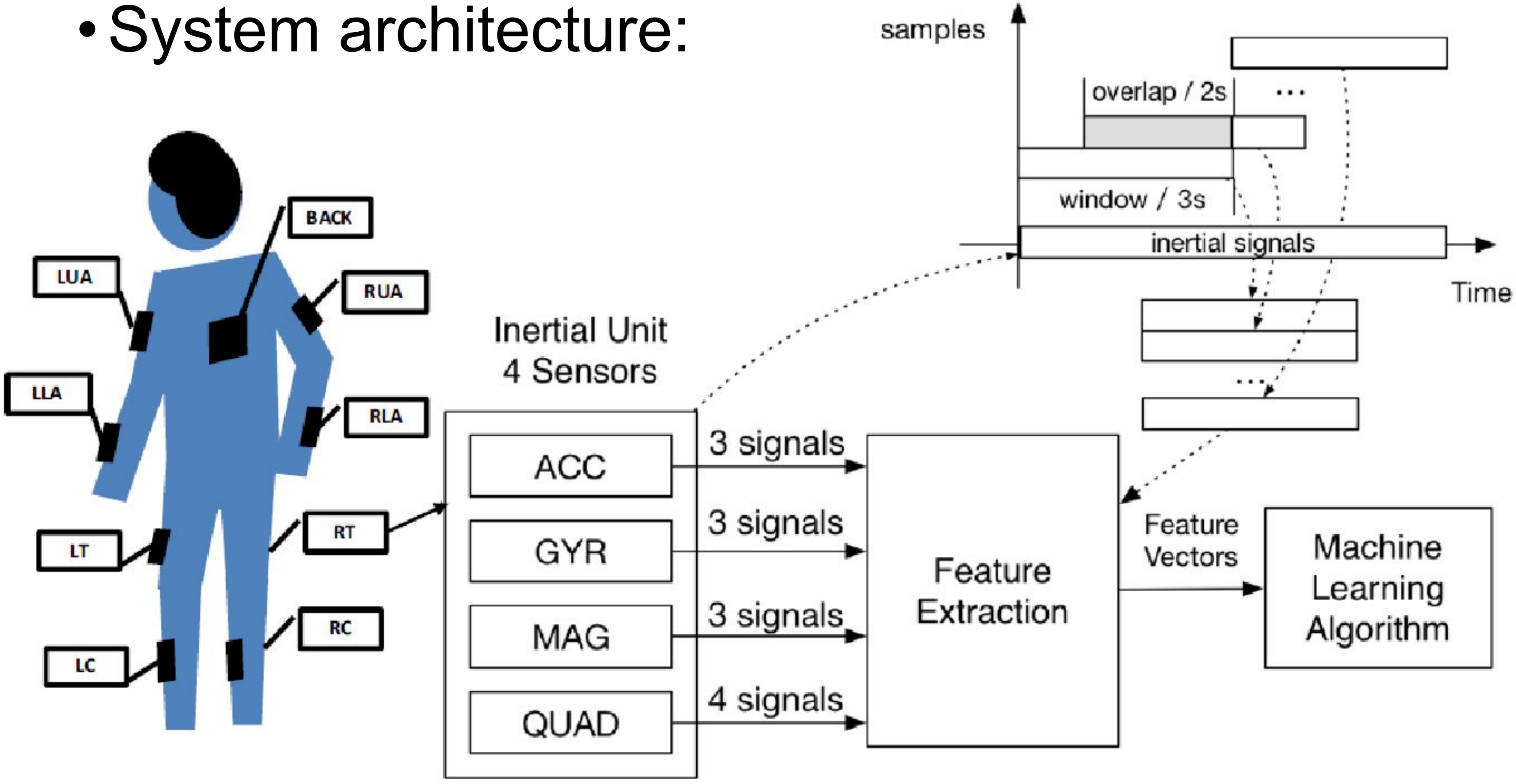
What is it?	How does it help?
• **Embedded intelligence** is the ability of a product, process or service to monitor its - Operational performance, - usage load, - environment • Goals: - enhance performance and lifetime, - increase quality and - improve customer satisfaction	• Monitor health and usage of products to extend their performance and lifetime • Improve market appeal and acceptance of products • The ability for a service, system or product to be used by ageing and people with special needs • Address skills shortages in limited resources • Enabling new revenue opportunities

Example: Architecture of System



Health monitoring: Physical activity recognition

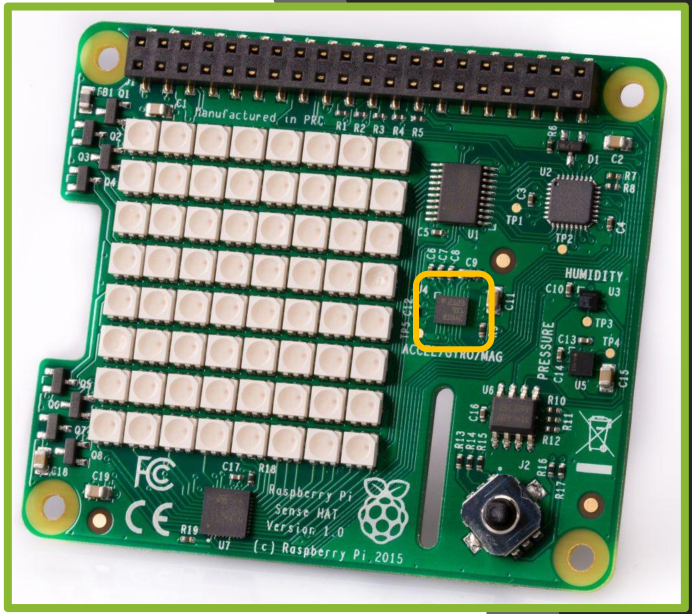
- System architecture:



Health monitoring: Demo

Contents

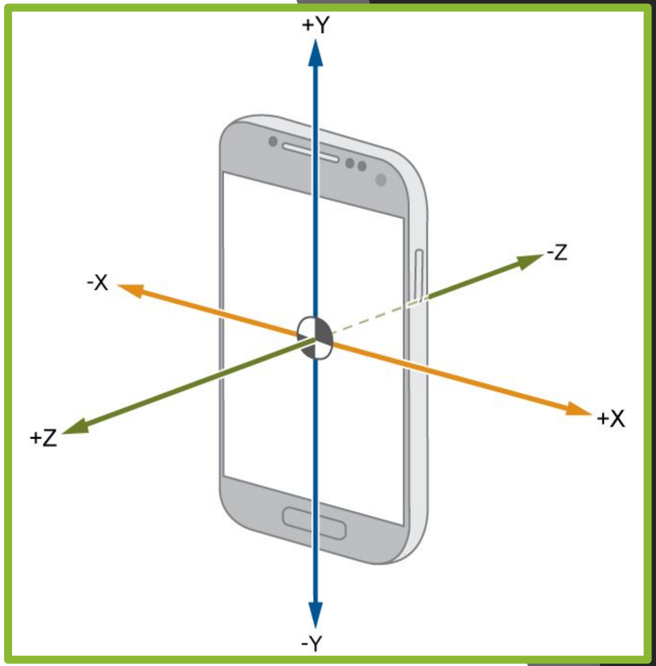
Sense HAT



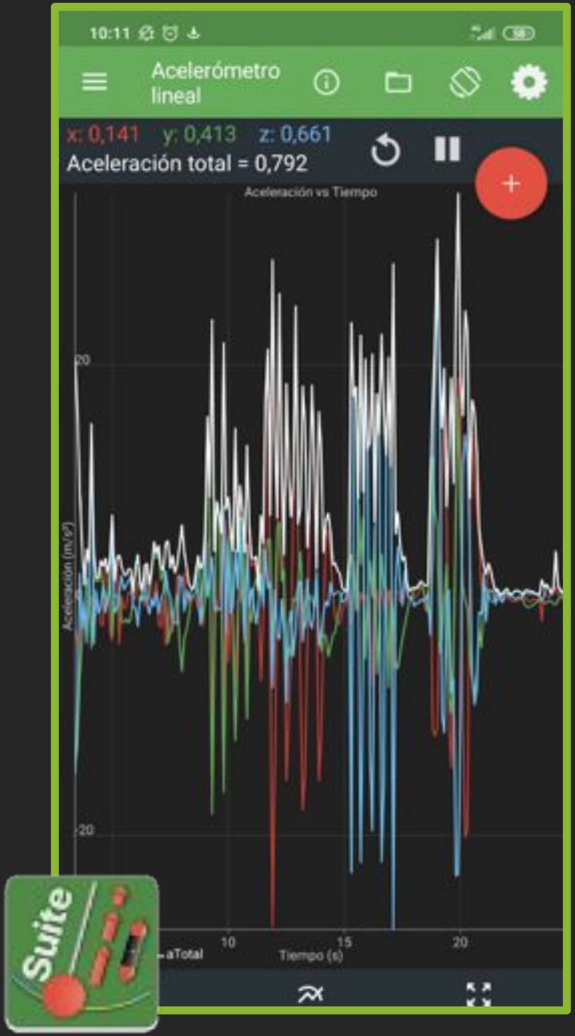
- Add-on board for Raspberry Pi
  - Made especially for the Astro Pi mission
- 8×8 RGB LED matrix
- Five-button joystick
- Sensors:
  - Gyroscope
  - **Accelerometer**
  - Magnetometer
  - Temperature
  - Barometric pressure
  - Humidity
- Python library

Departamento de Ingeniería Electrónica

Smartphone



- Sensors:
  - Gyroscope
  - **Accelerometer**
  - Magnetometer
  - Camera
  - …

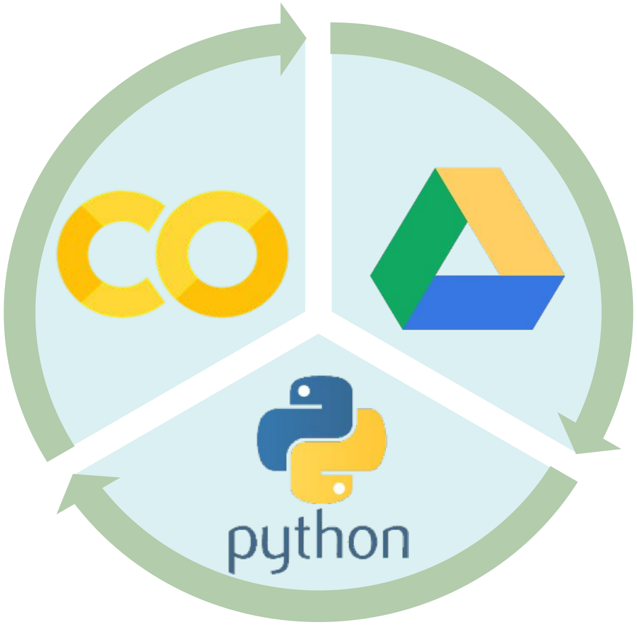


Contents

Software tools

Colab allows you to write and execute Python in your browser with:

- No configuration
- Free access to GPUs
- Easy sharing



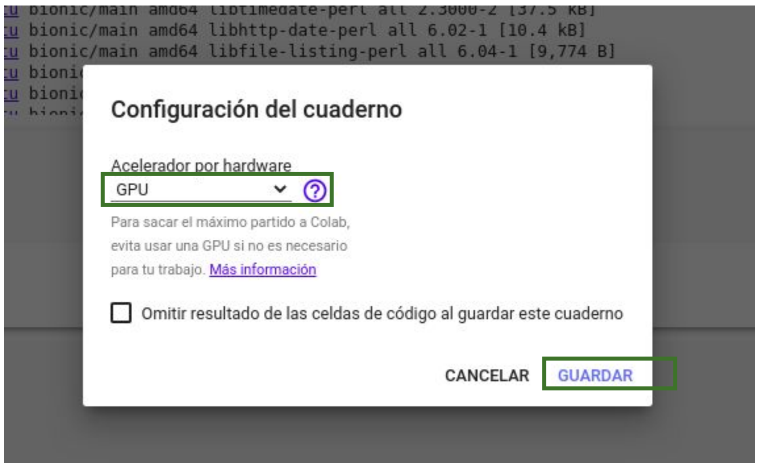
Drive: file storage and synchronization service developed by Google

Python: is an interpreted, high-level and general-purpose programming language

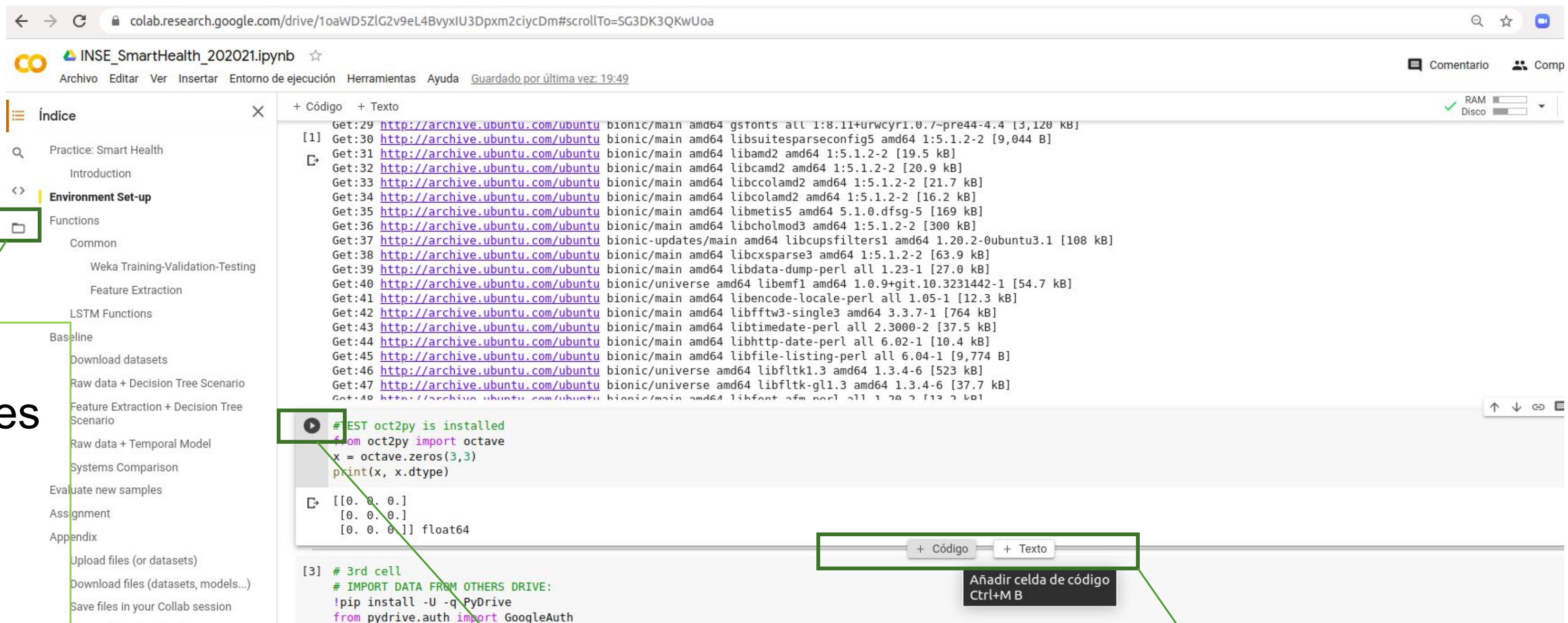
1) **Download this**
2) **Upload it to your personal Google Drive account**
3) **Double-click on it to launch Colab**

Introduction to Colab
Setting GPU environment

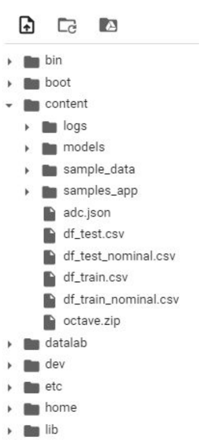
https://colab.research.google.com/



Introduction to Colab



List of folders/files



Run a code cell

Create new cell (Text or Code Cell)

Software architecture

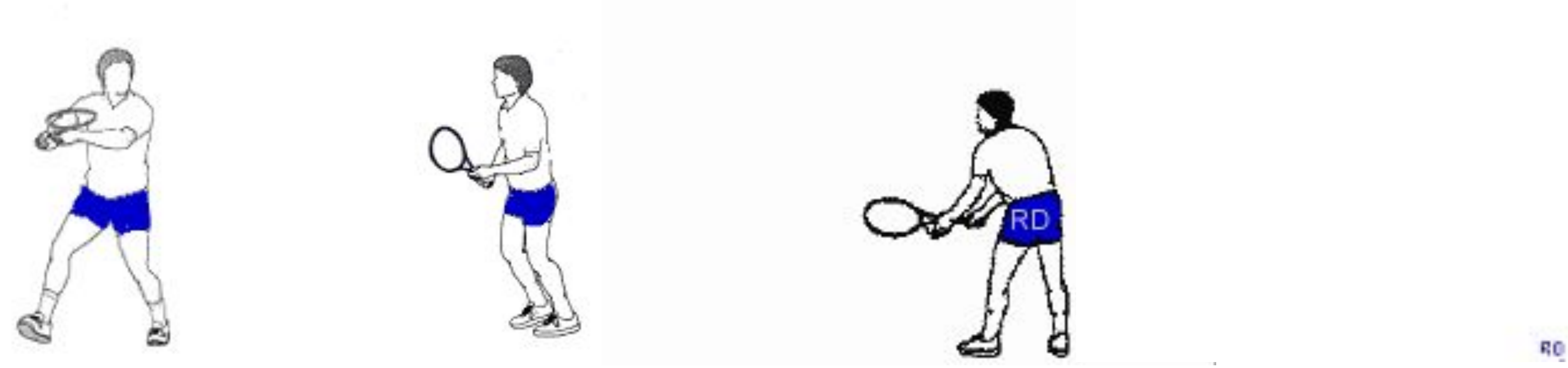
- Main colab script:
  - Practice SmartHealth
- Project scenarios:
  - Raw data + Decision Tree
  - Feature Extraction + Decision Tree
  - Raw data + Temporal Model

Índice

- **Practice: Smart Health**
  - Introduction
- Environment Set-up
- Functions
  - Weka Training-Validation-Testing
  - Feature Extraction
  - LSTM Functions
  - Common
- Baseline
  - Download datasets
  - Raw data + Decision Tree Scenario
  - MFCC Feature Extraction + Decision Tree Scenario
  - Raw data + Temporal Model (LSTM)
  - Systems Comparison
- Assignment
- Appendix
  - Upload files (or datasets)
  - Download files (datasets, models...)
  - Save files in your Collab session
  - Save and Load networks
- Dataset as ARFF
- Physics Toolbox Suite

Gesture recognition

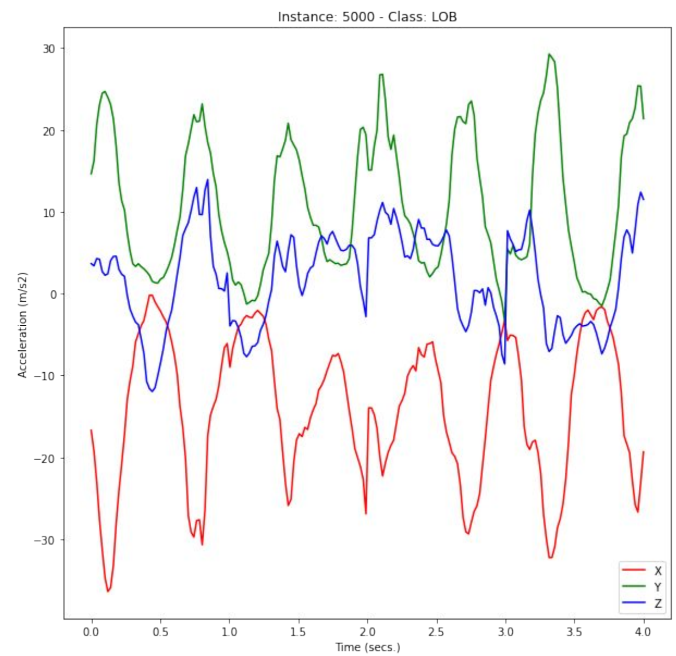
- Basic (table ☺) tennis techniques or strokes:



Drive (1) Backhand (2) Serve (3) Lob (4)

Gesture recognition

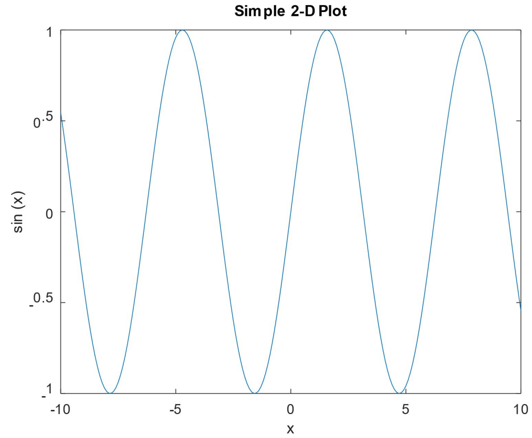
- Based in a 3D accelerometer
  - **Windowed analysis** of acceleration values
    - Different window lengths
    - Overlapped windows
  - Low sampling frequency: 50 Hz
- Simple front-end:
  - Only 60 mfcc-plp features
    - 20 features per channel
  - 6 functionals (mean & stdev)
  - Classifiers: Random Forest, LSTM



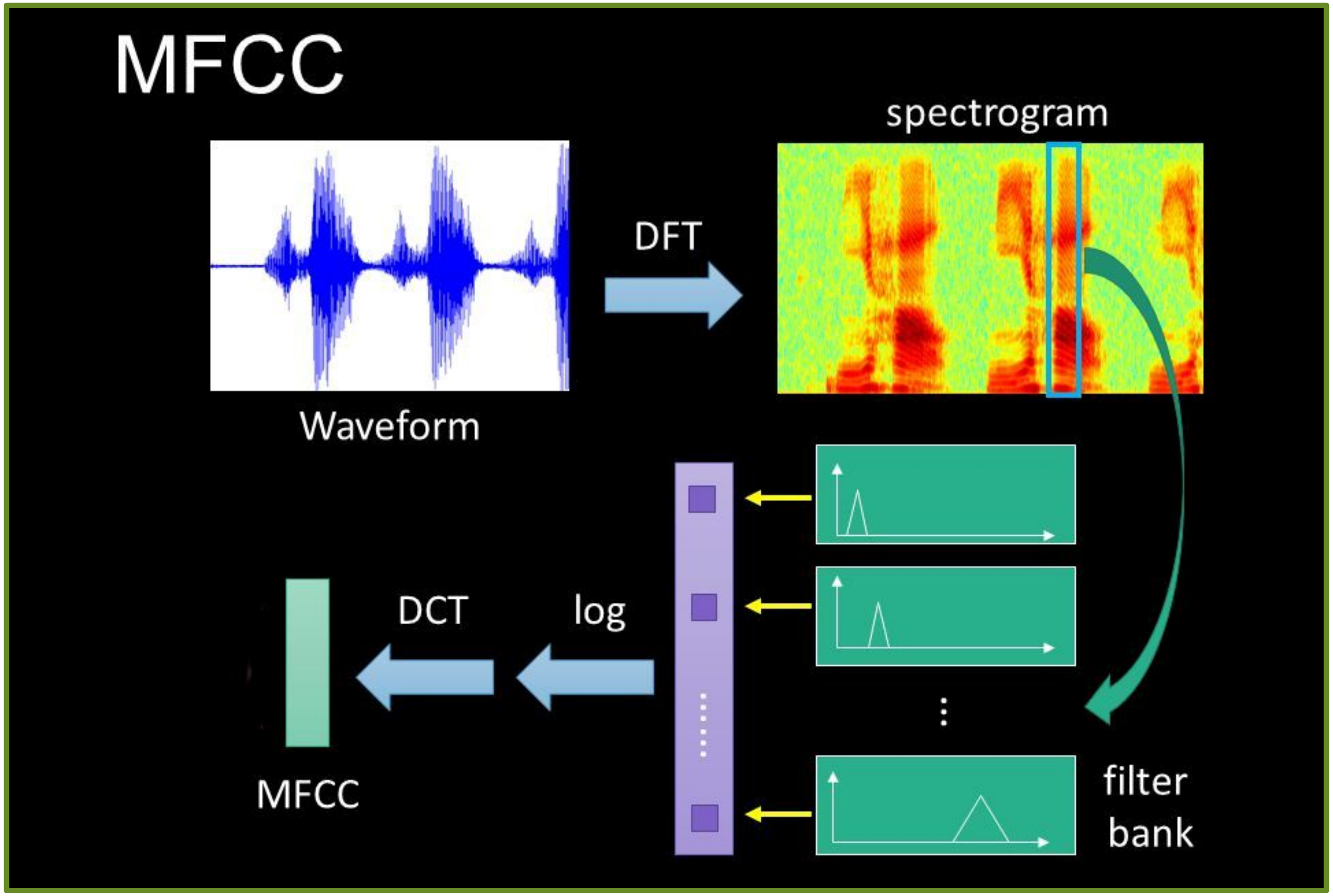
Feature Extraction

- Scientific Programming Language
- Powerful mathematics-oriented syntax with built-in plotting and visualization tools
- Free software, runs on GNU/Linux, macOS, BSD, and Windows
- Drop-in compatible with many Matlab script

```
# Create an evenly-spaced vector from -10..10
x = -10:0.1:10;
y = sin(x); # y is also a vector
plot(x, y);
title("Simple 2-D Plot");
xlabel("x");
ylabel("sin (x)");
```



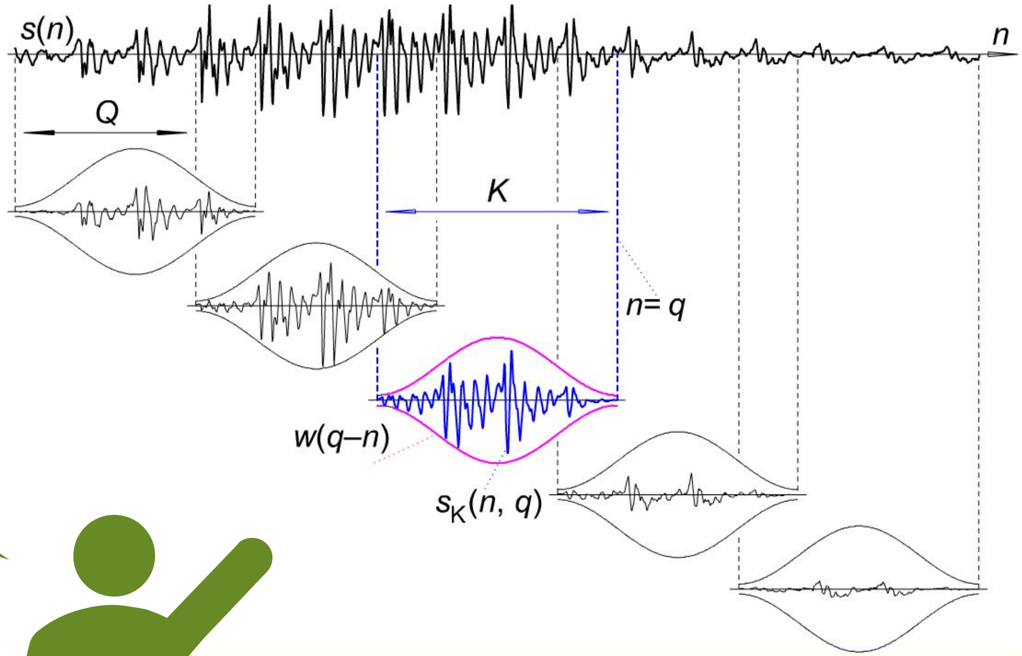
Feature Extraction (2)



Feature Extraction (3)

```
FeatureExtraction (AccX array,AccY array,AccZ_array(
     200, # WINDOW SIZE: K (4 seconds)
     150, # OVERLAP: K – Q (3 seconds)
     False)
```

Windowing is a means to **stationarize** signals. Tools like **Fourier-based techniques** are well-suited to stationary signals.



Feature Extraction (4)

GNU Octave

```
from oct2py import octave # Oct2Py: Python to GNU Octave Bridge
def FeatureExtraction (AccX_array,AccY_array,AccZ_array,
window_size,overlap,debug):
    aux_AccX_array = AccX_array[np.newaxis, :].T
    aux_AccY_array = AccY_array[np.newaxis, :].T
    aux_AccZ_array = AccZ_array[np.newaxis, :].T
    ...
    mfcc_plp = octave.calcula_features_mfcc_plp_online_new(aux_AccX
aux_AccY_array,          aux_AccZ_array, window_size, overlap, False)
%30plp+30mfcc
    functionals = [np.mean(aux_AccX_array),…,np.std(aux_AccX_array),…]
    %30plp+30mfcc+6functionals=66 features)
    res = np.append((np.asarray(mfcc_plp, …)), (np.asarray(functionals,…)))
```

Oct2py runs m-files from python

```
/usr/local/lib/octave/calcula_features_mfcc_plp_online_new.m
function OUTPUT =
calcula_features_mfcc_plp_online_new (AccXorig,AccYorig,AccZorig,
win_size,overlap)
 ...
 OUTPUT = frastaplp3D(AccX',AccY',AccZ', win_size, step);  %30 features
(10+10+10)
 OUTPUT = [OUTPUT fmfcc3D(AccX',AccY',AccZ', win_size, step)];  %30 features
(idem)
```

Generated dataset

dataset_test_mfcc

...	X_PLP_param1	X_PLP_param2	X_PLP_param3	X_PLP_param4	X_PLP_param5	...	Z_MFCC_param60	X_mean_param	Y_mean_param	Z_mean_param	X_std_param	Y_std_param	Z_std_param	user	class
0	27.958436	1.792896	0.926759	0.278980	0.390426		0.974356	6.562834	6.638258	0.282533	2.511517	3.209461	3.624661	15	DRIVE
1	28.394158	1.744596	0.834124	0.150108	0.108082		-0.998596	5.054386	6.178057	-1.635454	5.180306	6.246923	5.153387	15	DRIVE
2	28.271179	1.223328	1.108851	0.755840	0.342047		-0.809730	4.535134	6.343756	-2.171127	5.488920	7.083468	5.056593	15	DRIVE
3	29.710130	1.286819	-0.219463	0.771217	-0.053580		-2.450957	4.061923	6.211137	-0.709630	5.387086	7.425837	6.012131	15	DRIVE
4	29.605985	1.300449	-0.039957	0.903575	0.167716		-1.657046	3.639123	6.371750	1.139097	5.470489	8.576052	7.393074	15	DRIVE
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3835	28.423973	1.272255	0.619025	0.892203	-0.743771		-1.966657	-2.958946	4.502018	5.495929	4.909814	3.949686	7.236630	20	LOB
3836	28.460169	1.431241	0.651814	0.437103	-0.689868		-2.727839	-2.926883	4.677041	5.925444	4.919016	4.100518	6.692011	20	LOB
3837	28.448223	1.415562	1.002449	0.808588	-0.635030		-0.990033	-4.033475	5.615216	6.889087	4.730290	4.327907	6.212749	20	LOB
3838	29.302487	1.041828	0.690837	0.819883	-0.069165		0.479920	-4.806968	6.434846	7.822439	4.194530	4.210162	5.109674	20	LOB
3839	29.445724	0.845920	0.617501	0.816422	-0.001756		0.715593	-5.614426	7.062794	8.811072	3.497779	3.999955	3.995048	20	LOB

Create Model Weka

```
/Functions/Weka Training-Validation-Testing
from weka.classifiers import Classifier
def CreateClassifier(classname,options):
  model = Classifier(classname=classname, options=options)
  return model
```

```
/Feature Extraction + Decision Tree Scenario
options_RandomForest
=["-P","100", "-I", "100", "-num-slots","1", "-K", "0","-M","1.0", "-V", "
0.001", "-S", "1"]
model_arquitecture_mfcc = CreateClassifier(
     classname="weka.classifiers.trees.RandomForest" ,
     options=options_RandomForest )
```

Create Model TF

```
/Functions/LSTM Functions
from weka.classifiers import Classifier
def CreateLSTM(neurons_first_layer,n_steps,dimension,loss,optimizer,metrics):
  model_lstm = Sequential()
  model_lstm.add(LSTM(neurons_first_layer, input_shape=(n_steps, dimension),
  return_sequences=False)
  model_lstm.add(Dense(4, activation='softmax'))
  model_lstm.compile(loss=loss, optimizer=optimizer, metrics=metrics)
  return model_lstm
```

```
/Raw data + Temporal Model
model_lstm_ES = CreateLSTM(neurons_first_layer = 200, n_steps = 200, dimension =
 3, loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])
```

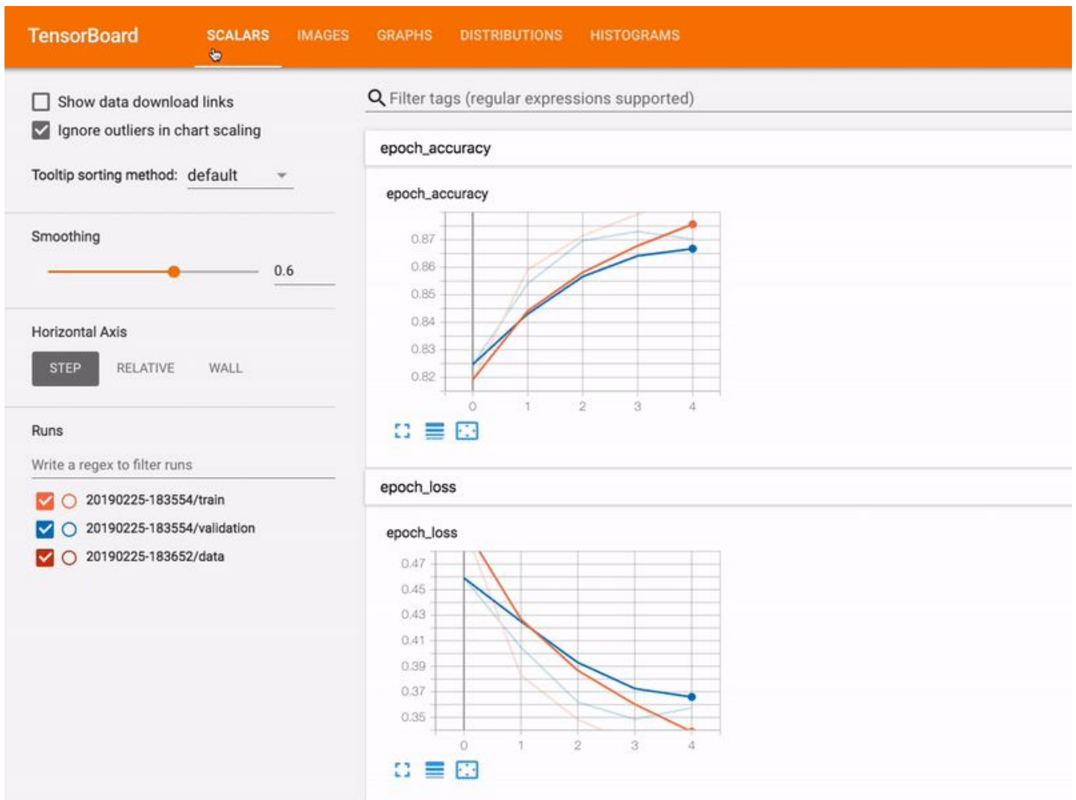
Train Model Weka

```
/Functions/Weka Training-Validation-Testing
def TrainModel(train_dataset, classifier, output_path_model):
    #Indicate where is class label
    train_dataset.class_is_last()
    # train classifier
    classifier.build_classifier(train_dataset)
    # save classifier object
    print("\n[TrainedModel]\n\n")
    serialization.write(output_path_model, classifier)
    return classifier
```

```
#Train and save model
out_path_model_mfcc = "models/trained_TreeRandomForest_mfcc.model"
trained_model = TrainModel(dataset_train_mfcc, model_arquitecture_mfcc, out_path
_model_mfcc)
```

Train Model TF

```
#Train model
model_lstm_ES.fit(X_train,y_train,batch_size= 16,epochs=10,
               validation_data=(X_validation, y_validation),…)
```



Recognition Weka

```
/Functions/Weka Training-Validation-Testing
def Recognize(trained_classifier,test_dataset):
    print("# - actual - predicted - error - class distribution" )
    for index, inst in enumerate(test_dataset):
      pred = trained_classifier.classify_instance(inst)
      dist = trained_classifier.distribution_for_instance(inst)
      print("%d - %s - %s - %s  - %s"  %
                (index+1,
                inst.get_string_value(inst.class_index),
                inst.class_attribute.value(int(pred)),
                "yes" if pred != inst.get_value(inst.class_index)  else "no",
                str(dist.tolist())))
```

```
#SEE PREDICTIONS:
Recognize(trained_model, dataset_test_weka)
```

Recognition TF

/Functions/Weka Training-Validation-Testing

```
def Recognize_LSTM(trained_LSTM,test_parameters,test_labels):
  y_pred = trained_LSTM.predict(test_parameters)
  print("# - actual - predicted - error - class distribution" )
  for index in range(0,len(test_labels)):
    pred = Convert2nominal_pingPong (np.argmax(y_pred[index],axis= -1)+1)
    true_val = Convert2nominal_pingPong (np.argmax(test_labels[index],axis= -1)+1
)
    print("%d - %s - %s - %s  - %s" %
                    (index+ 1,
                    true_val,
                    pred,
                    "yes" if pred != true_val else "no",
                    str(y_pred[index])))
```

```
#SEE PREDICTIONS:
Recognize_LSTM(model_lstm_ES,X_test,y_test)
```

Evaluation Weka

```
/Functions/Weka Training-Validation-Testing
from weka.classifiers import Evaluation
def EvalClassifier(trained_classifier, train_dataset, test_dataset):
    evl_train = Evaluation(train_dataset)
    evl_test = Evaluation(train_dataset)
    print("--------RESULTS IN TRAINING----------" )
    evl_train.test_model(trained_classifier, train_dataset)
    print(evl_train.summary())
    print("--------RESULTS IN TEST----------" )
    evl_test.test_model(trained_classifier, test_dataset)
    print(evl_test.summary())
```

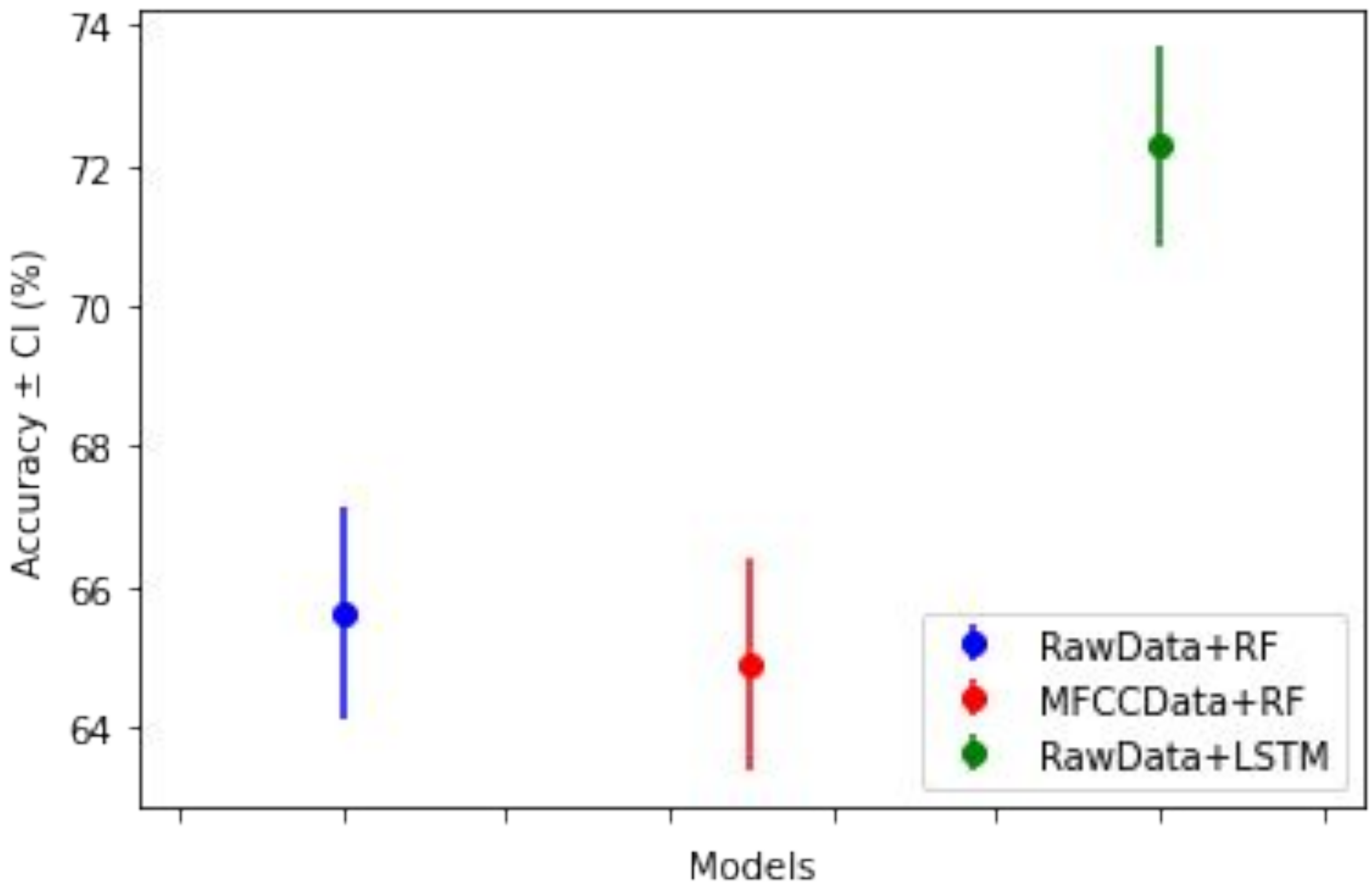
```
#EVAL TRAINED MODEL WITH MFCC FEATURES
EvalClassifier(trained_model, dataset_train_mfcc, dataset_test_mfcc)
```

Evaluation TF

```
/Functions/Weka Training-Validation-Testing
from weka.classifiers import Evaluation
def EvalLSTM(y_labels, y_pred):
  y_test_df = pd.DataFrame({ 0:y_labels[:,0],1:y_labels[:,1],2:y_labels[:,2],3:
y_labels[:,3]})
  y_test_df_idx=y_test_df.idxmax(axis= 1)
  print(confusion_matrix(y_test_df_idx, y_pred))
  print(classification_report(y_test_df_idx, y_pred))
  print('Accuracy %0.4f' % accuracy_score(y_test_df_idx, y_pred))
```

```
#Eval model on training/test sets
print("---- TRAINING RESULTS: -----" )
EvalLSTM(y_train, y_pred_train_ES)
print("---- TEST RESULTS: -----" )
EvalLSTM(y_test, y_pred_test_ES)
```

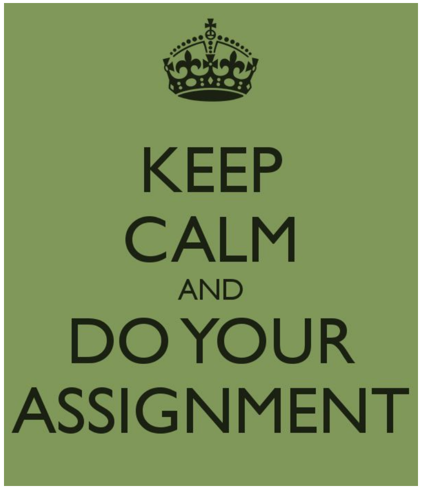
Systems Comparison



Contents

Assignment

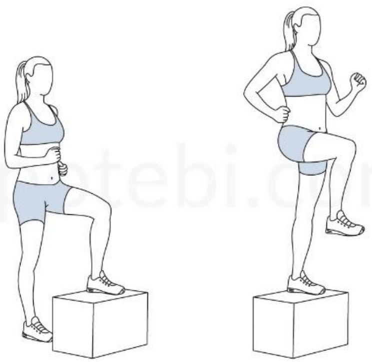
- STANDARD: **Re-run the notebook for a different task**
  - New dataset: INSE1920-challenge-ASSIGNMENT.csv
  - **Create new train and test partitions**
  - **Try a different Weka classifier**: Random Forest, NB, SVM, NN, ...
  - Add a paragraph (text cell) at the end briefly **summarizing/discussing the obtained results**
    - (captions are allowed)
  - **Save the .ipynb notebook and submit it to** fernando.fernandezm@upm.es with the following subject: **INSE-SmartHealth-Team_XX-Pepito_y_Paquito**



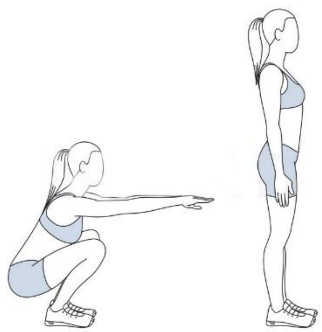
New dataset: classes

Running in place

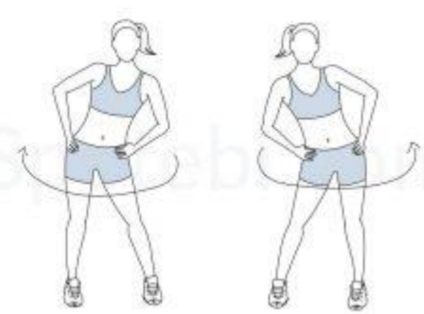
Step up with one knee raise



Squat

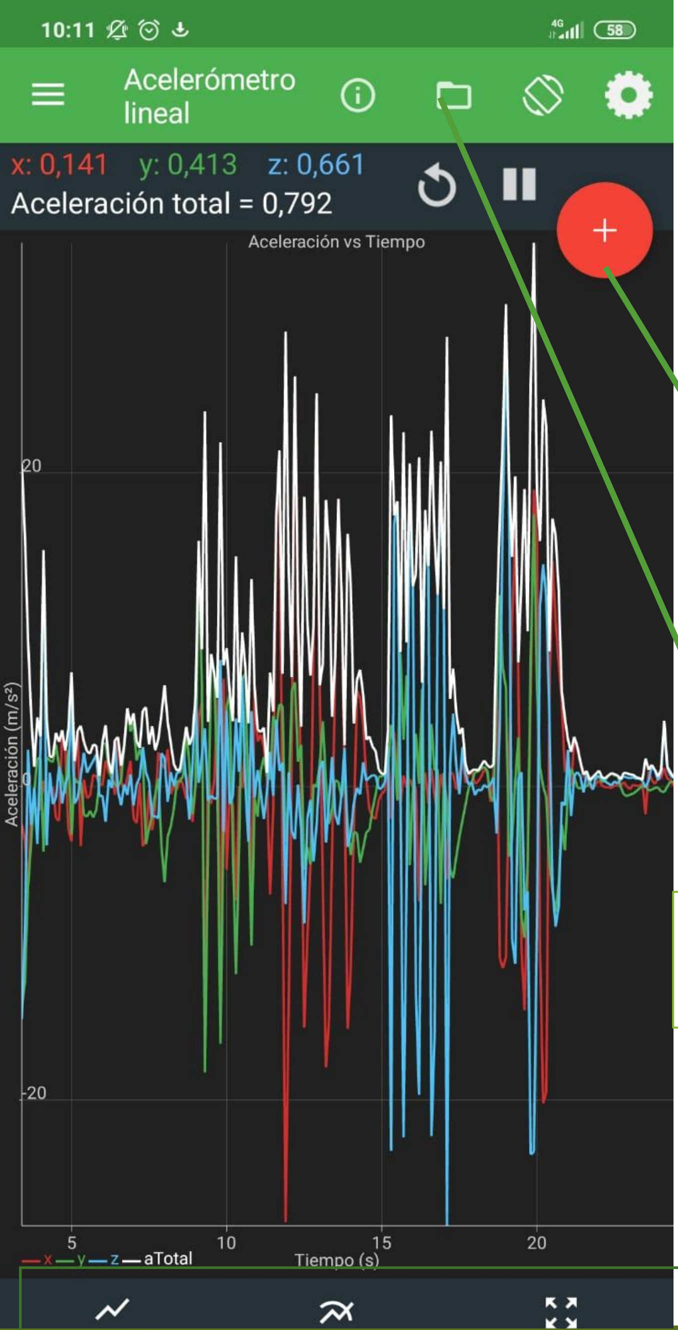


Hip circles



Alternative assignment (OPTIONAL)

- **Design your own task** by defining a new inventory of gestures:
  - Should be **repetitive movements** (to simplify the recording)
  - For example, “swimming” skills:
    - Backstrokes, Front crawl, Breaststroke, Butterfly, …
- **Record your own dataset** at home using the “Suite” app
  - Android and iOS versions available
  - Returns a .csv for every recording session
  - Additional libraries available (sampling rate and windowing)
  - At least 1 minute per class, both members
  - Same device, same hand
- **Use the notebook to train and evaluate the system**



Get raw data

- **Physics Toolbox Suite:** Mobile application for recording raw data from smartphone sensors.
- It also let send/upload information to different platforms like Drive, Whatsapp…
- Sampling frequency: 200 Hz

Files managment

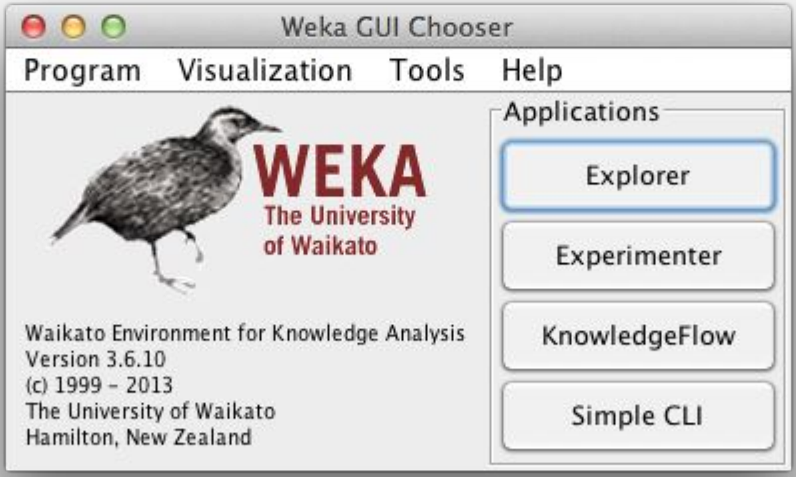
Start/Stop data recording

Plot customizazion options

Contents

Weka Experimenter

1. Start Weka

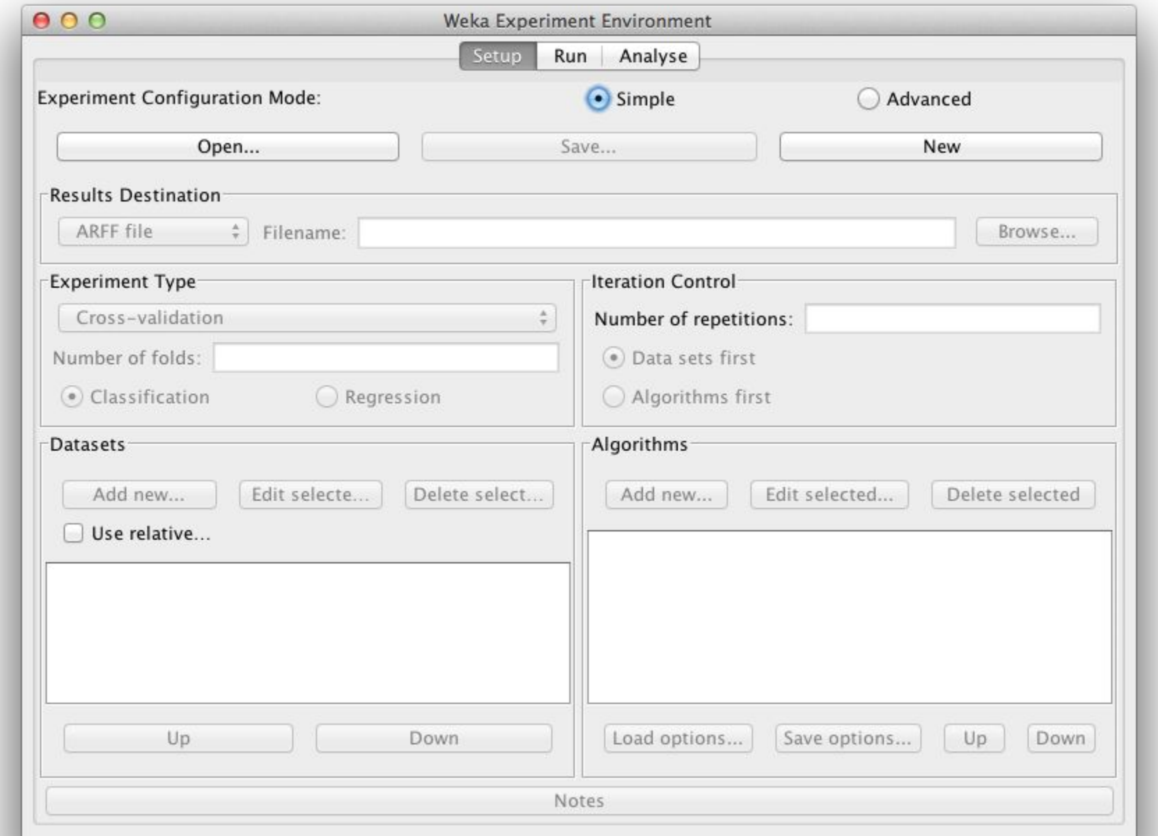


Click the “Experimenter” button to launch the Weka Experimenter.

The Weka Experimenter allows you to design your own experiments of running algorithms on datasets, run the experiments and analyze the results. It’s a powerful tool.

Weka Experimenter

2. Design Experiment



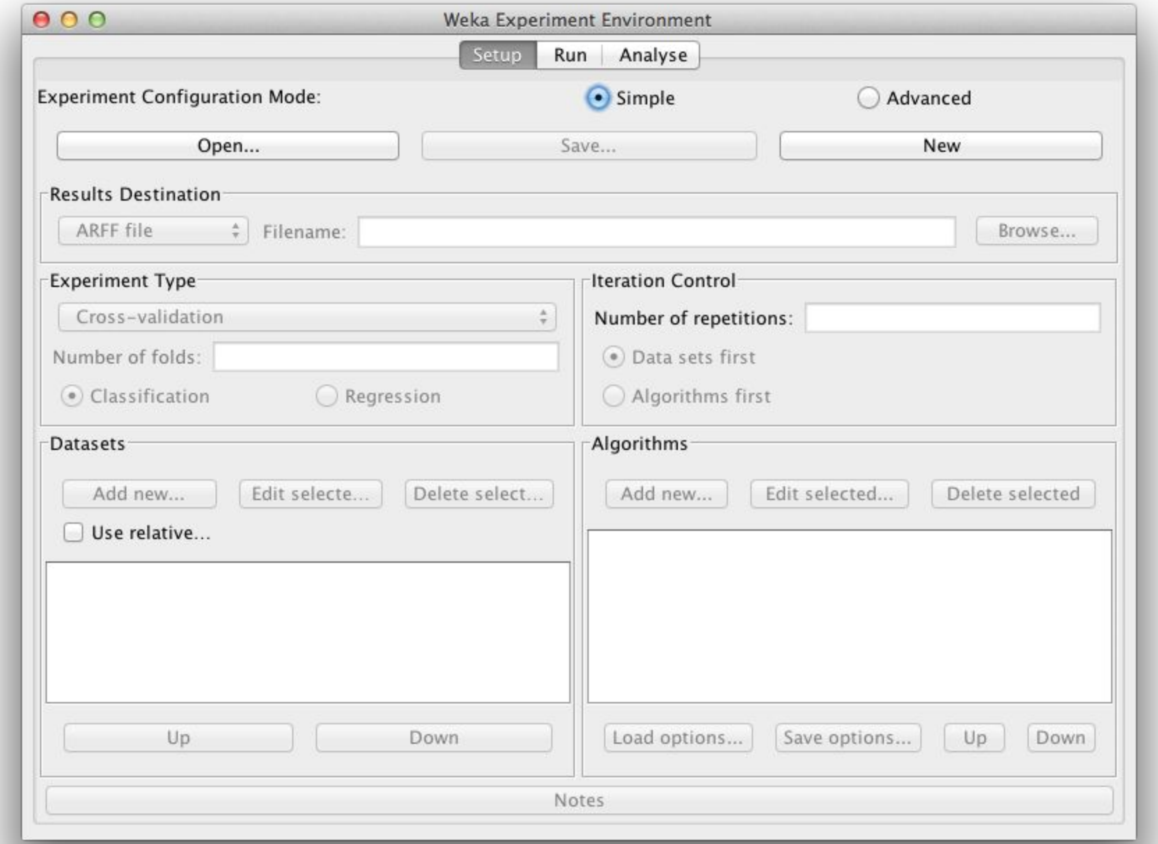
Click the "New" button to create a new experiment configuration.

Test Options

The experimenter configures the test options for you with sensible defaults. The experiment is configured to use Cross Validation with 10 folds. It is a "Classification" type problem and each algorithm + dataset combination is run 10 times (iteration control).

Weka Experimenter

2. Design Experiment



In the “Datasets” select click the “Add new…” button.

Open the “data” directory and choose the “aux_features_raspi.arff” dataset (“iris.arff” in the figure).

Now let’s choose 3 algorithms to run our dataset.

Weka Experimenter

- **ZeroR** is the simplest algorithm we can run. It picks the class value that is the majority in the dataset and gives that for all predictions. It is good to have ZeroR as a baseline that we demand algorithms to outperform.

Click "Add new…" in the "Algorithms" section. Click the "Choose" button. Click "ZeroR" under the "rules" selection.

- **OneR** is like our second simplest algorithm. It picks one attribute that best correlates with the class value and splits it up to get the best prediction accuracy it can.

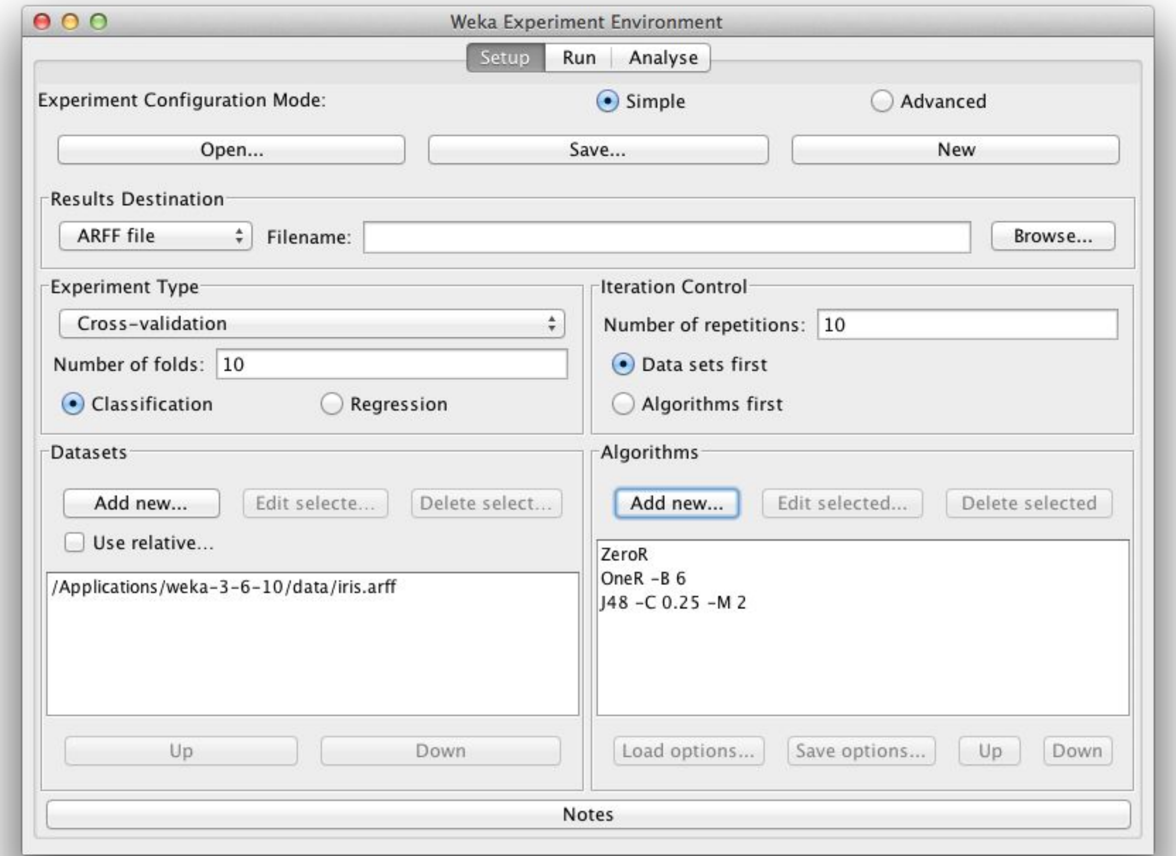
Click "Add new…" in the "Algorithms" section. Click the "Choose" button. Click "OneR" under the "rules" selection.

- **J48** is decision tree algorithm. It is an implementation of the C4.8 algorithm in Java ("J" for Java and 48 for C4.8).

Click "Add new…" in the "Algorithms" section. Click the "Choose" button. Click "J48" under the "trees" selection.

Weka Experimenter

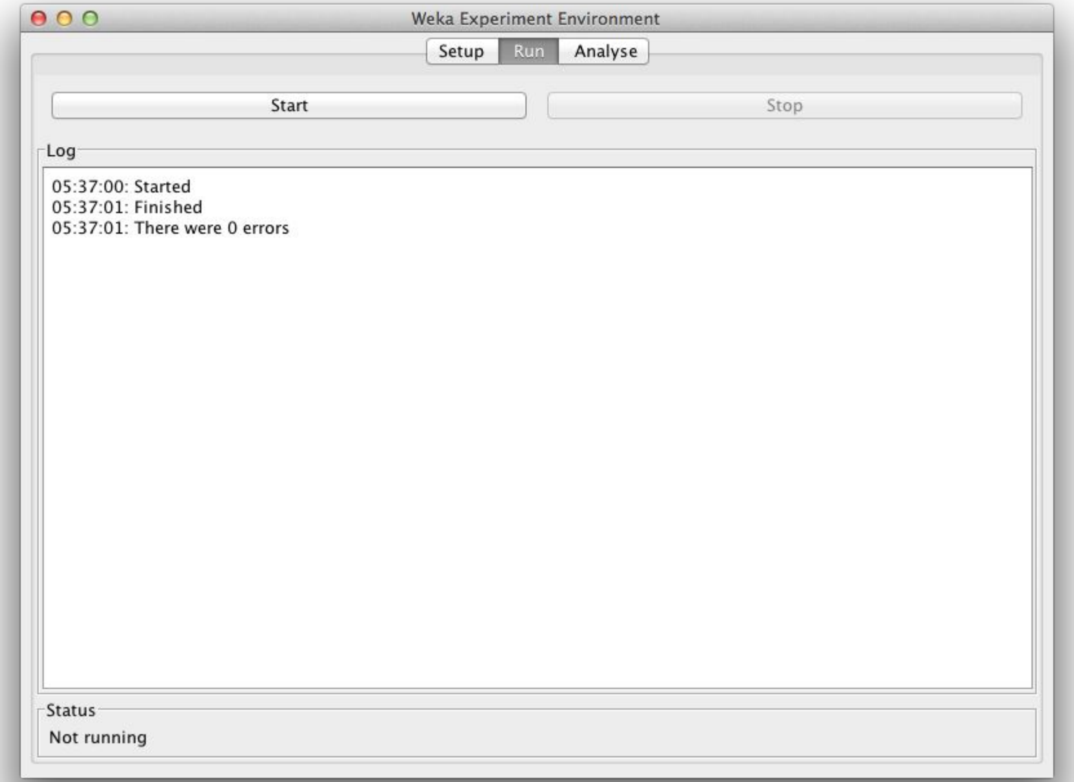
2. Design Experiment



We are ready to run our experiment!!!!

Weka Experimenter

3. Run Experiment



Click the “Run” tab at the top of the screen.

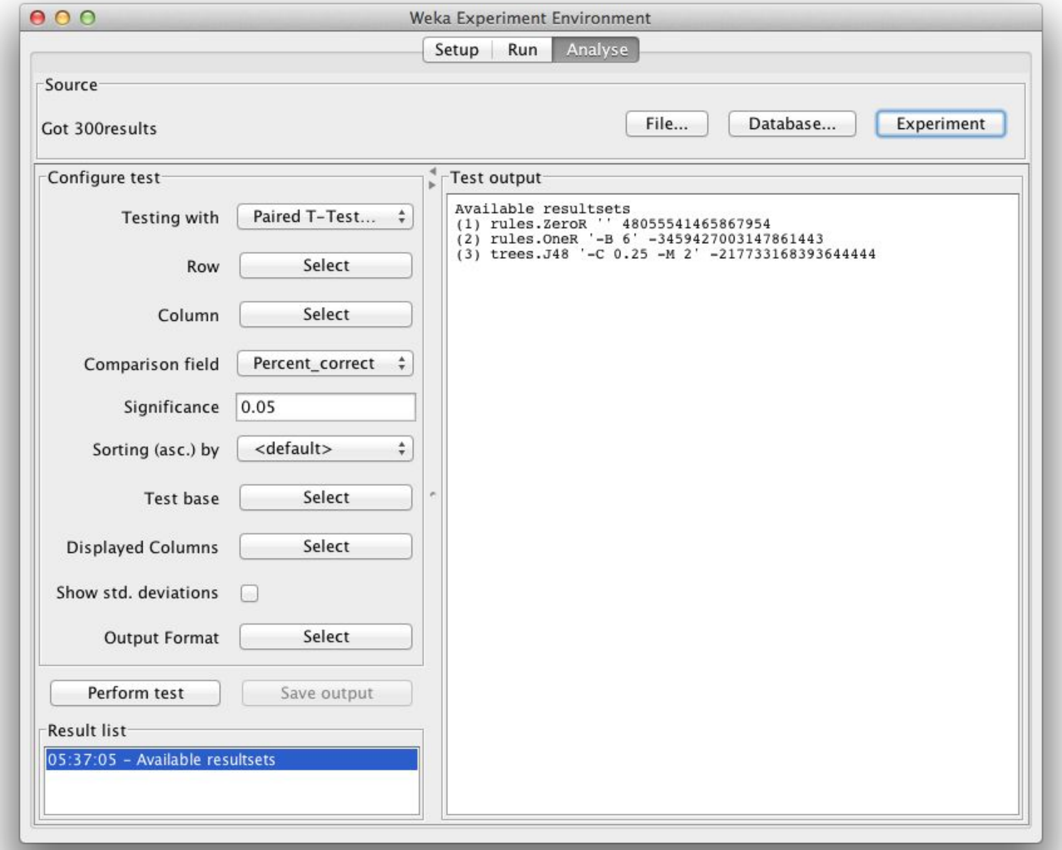
This tab is the control panel for running the currently configured experiment.

Click the big “Start” button to start the experiment and watch the “Log” and “Status” sections to keep an eye on how it is doing.

Given that the dataset is small and the algorithms are fast, the experiment should complete in seconds.

Weka Experimenter

4. Review Results



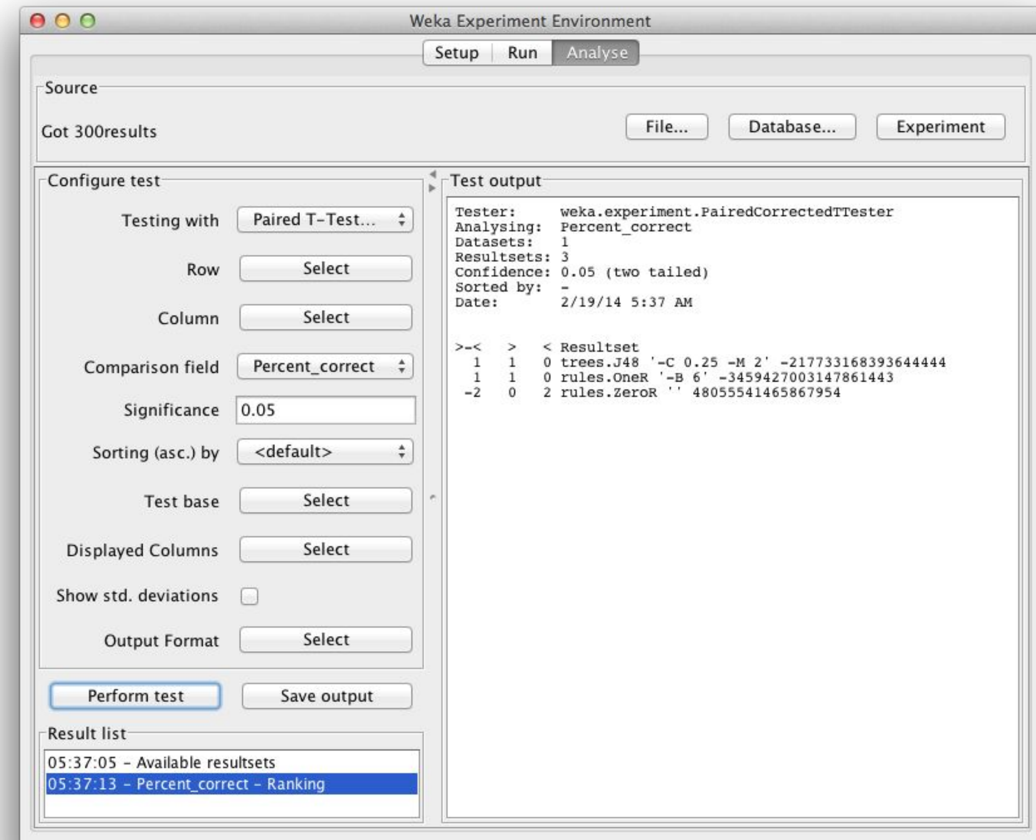
Click the “Analyse” tab at the top of the screen.

This will open up the experiment results analysis panel.

Click the “Experiment” button in the “Source” section to load the results from the current experiment.

Weka Experimenter

4. Review Results: Algorithm Rank



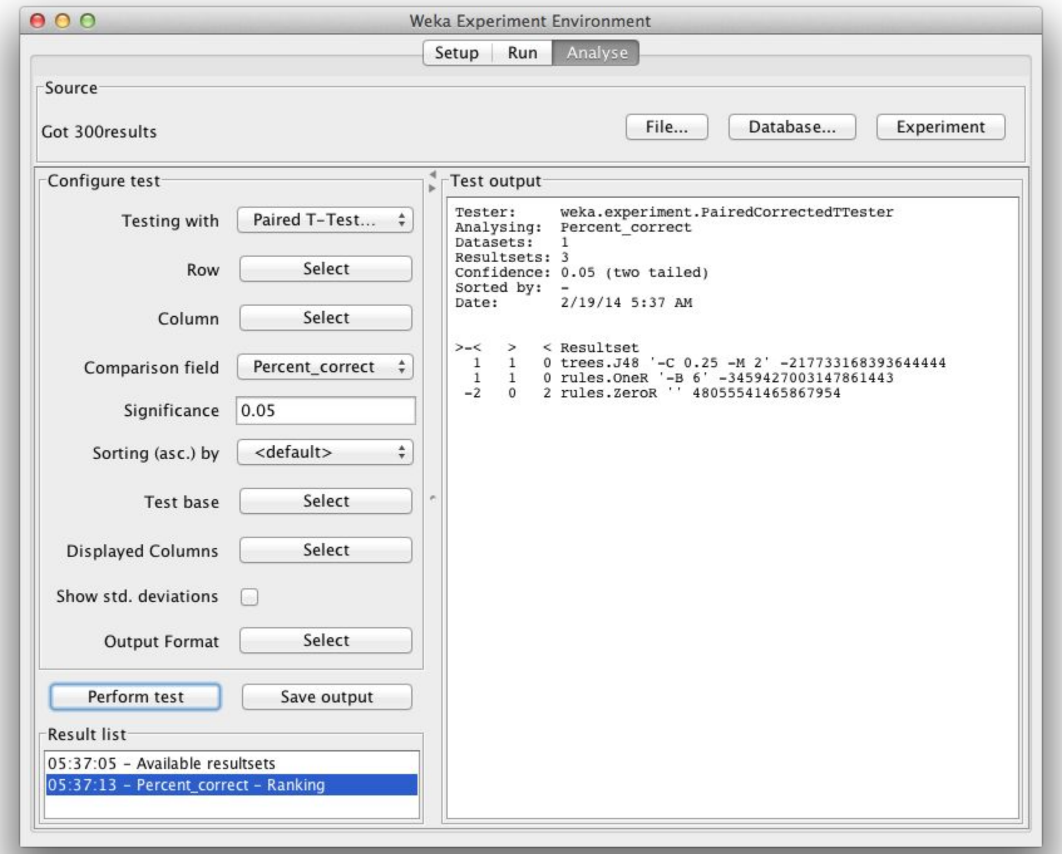
The first thing we want to know is which algorithm was the best. We can do that by ranking the algorithms by the number of times a given algorithm beat the other algorithms.

Click the “Select” button for the “Test base” and choose “Ranking“.

Now Click the “Perform test” button.

Weka Experimenter

4. Review Results: Algorithm Rank

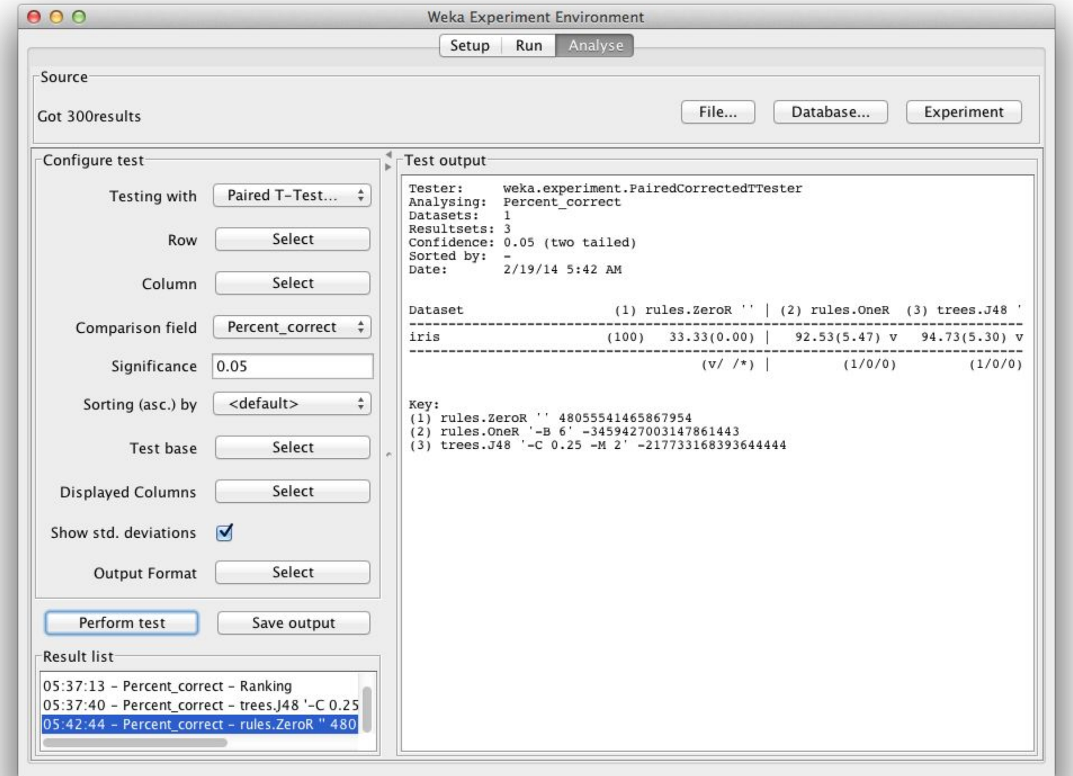


The ranking table shows the number of statistically significant wins each algorithm has had against all other algorithms on the dataset. A win, means an accuracy that is better than the accuracy of another algorithm and that the difference was statistically significant.

We can see that both J48 and OneR have one win each and that ZeroR has two losses. This is good, it means that OneR and J48 are both potentially contenders outperforming out baseline of ZeroR.

Weka Experimenter

4. Review Results: Algorithm Accuracy



Next we want to know what scores the algorithms achieved.

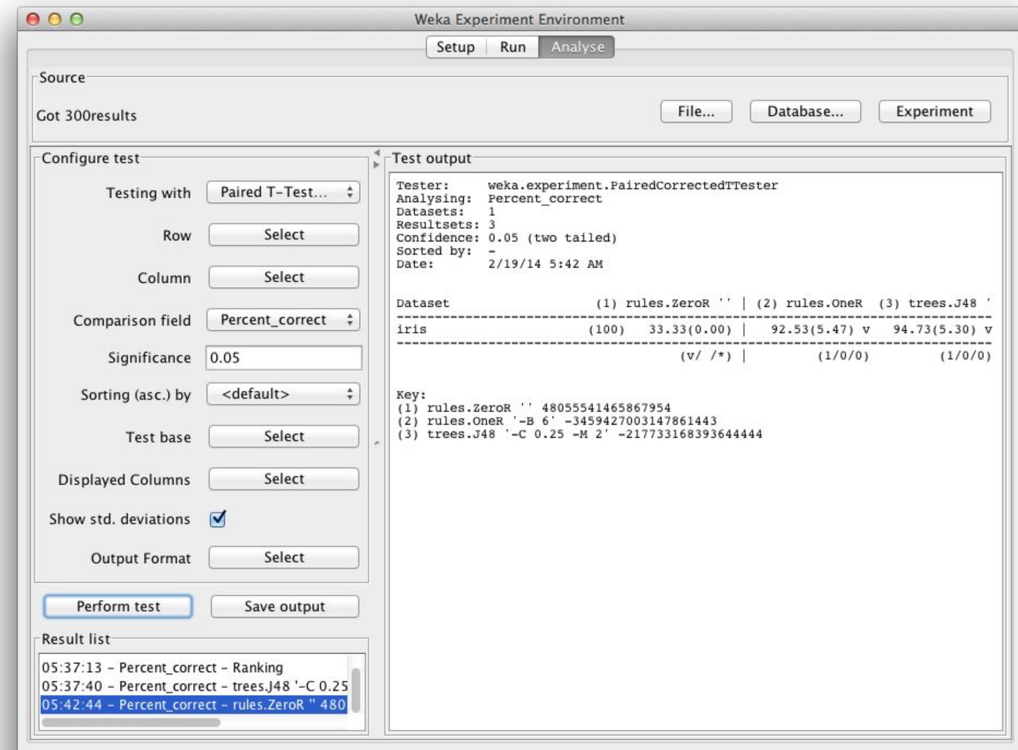
Click the "Select" button for the "Test base" and choose the "ZeroR" algorithm in the list and click the "Select" button.

Click the check-box next to "Show std. deviations".

Now click the "Perform test" button.

Weka Experimenter

4. Review Results: Algorithm Accuracy



In the "Test output" we can see a table with the results for 3 algorithms. Each algorithm was run 10 times on the dataset and the accuracy reported is the mean and the standard deviation in rackets of those 10 runs.

We can see that both the OneR and J48 algorithms have a little "v" next to their results. This means that the difference in the accuracy for these algorithms compared to ZeroR is statistically significant.

We can also see that the accuracy for these algorithms compared to ZeroR is high, so we can say that these two algorithms achieved a statistically significantly better result than the ZeroR baseline.

References

- Python to GNU Octave bridge
  https://pypi.python.org/pypi/oct2py
- Python wrapper for the Weka Machine Learning Workbench
  https://pypi.python.org/pypi/python-weka-wrapper
- WEKA in the Ecosystem for Scientific Computing
  http://www.cs.waikato.ac.nz/~eibe/WEKA_Ecosystem.ppt
- Introduction to Google Colab:
  https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=5fCEDCU_qrC0

Questions?

рахмат
danke
謝謝
ngiyabonga
Баярлалаа
spas
teşekkür ederim
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
dank je
misaotra
matondo
paldies
grazzi
спасибі
blagodaram
kiitos
dankie
nanni
nandri
dhanyavad
thank you
gracias
хвала
asante
manana
bedankt
bayarlalaa
gracie
hvala
maururu
koszonom
dankon
aciu
akun
obrigada
murakoze
tenki
enkosi
djiere dieuf
tau
mochchakkeram
дякую
mamnun
chokrane
dziękuję
chnorakaloutioun
gratias ago
gracies
sulpay
go raibh maith agat
sobodi
dekuji
taiku
obrigado
sagolun
sukriya
kop khun krap
arigatô
takk
dakujem
trugarez
mesi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
rahmet
terima kasih
kam sah hamnida
rahmat
ευχαριστώ
diolch
dhanyavadagalu
shukriya
merce
мерси
তোমাকে ধন্যবাদ
감사합니다
xiexie
merci