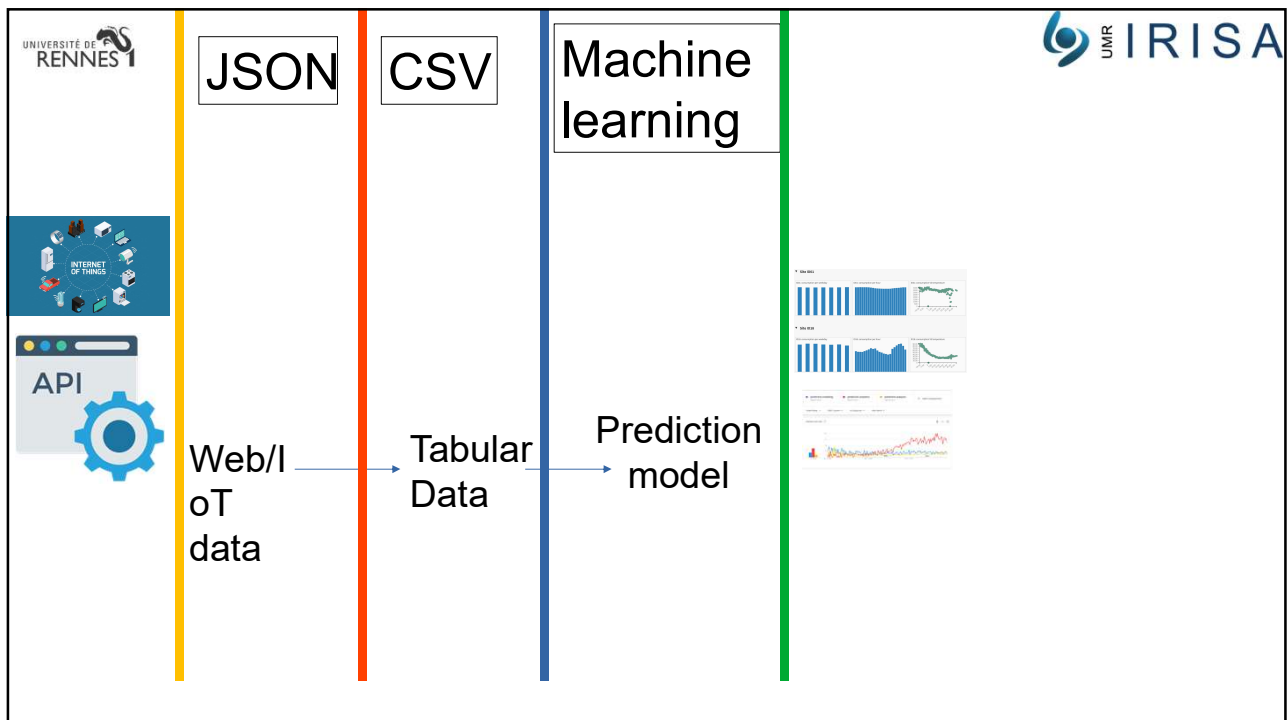
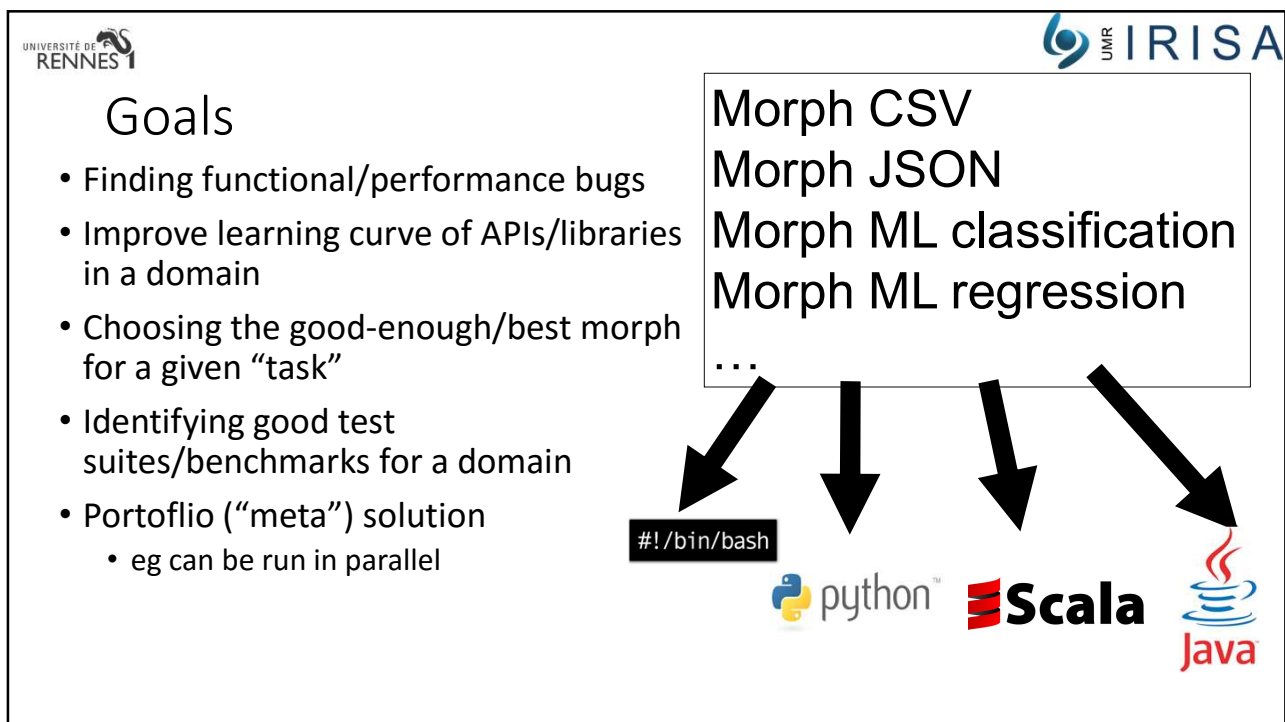
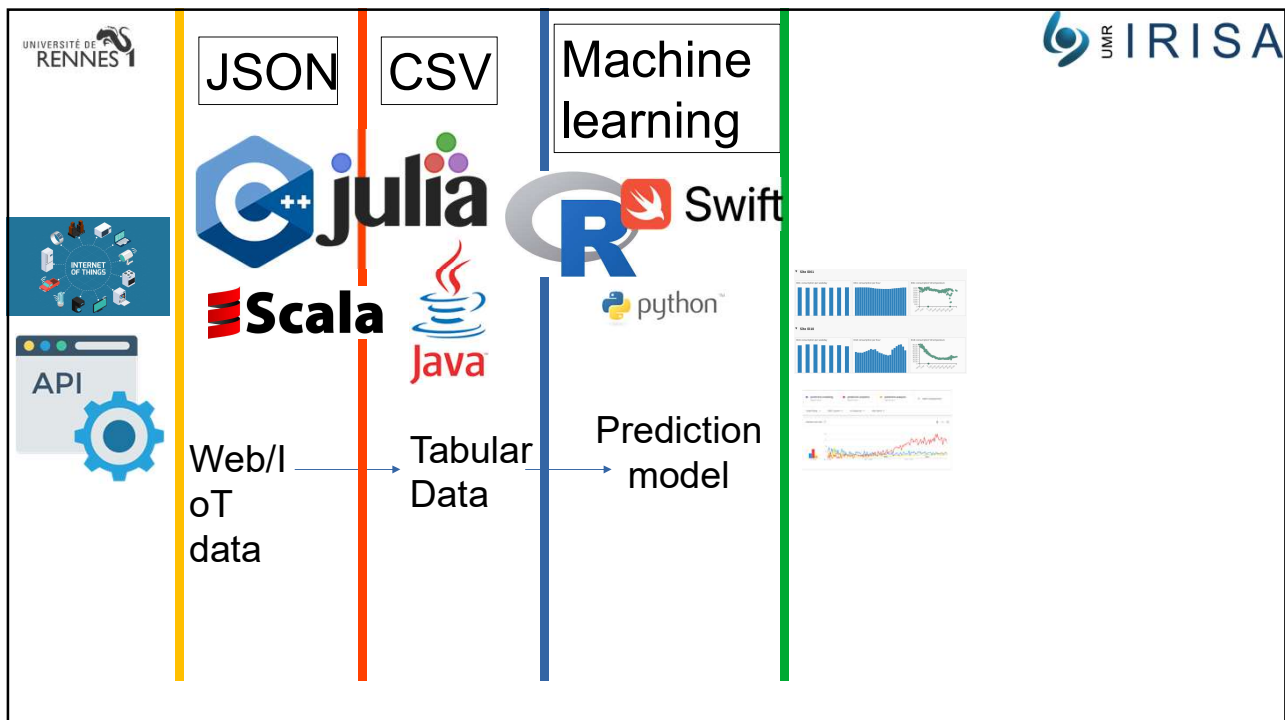


Master SIF / DSL

Project presentation

Jean-Marc Jézéquel / Mathieu Acher





DSL for JSON

- Concepts
 - Load, Store JSON files
 - Select subset of objects, Projection (slice of objet)
 - Aka core relational algebra operators
 - Compute basic \sum , \prod over fields
 - Print field value, #objects, #fields, depth, expressions...
 - Insert/modify/remove object/fields
- Services
 - Export to CSV
 - Interpreter
 - Compilers to (Java | Python | Julia) + JQ (for relevant subset)
 - <https://stedolan.github.io/jq/>

DSL for CSV

- Concepts
 - Load, Store CSV files
 - Select subset of lines/column (cut)
 - Aka core relational algebra operators
 - Compute basic \sum , \prod over fields
 - Print field value, #objects, #fields, expressions...
 - Insert/modify/remove lines/fields
- Services
 - Export to JSON
 - Interpreter
 - Compilers to (Java | Python | Julia) + bash (grep/cut/awk...)

Polymorphic CSV

```
f1 = "/tmp/test.csv"
n1 = #rows f1
```

```
wc -l /tmp/test.csv
```

```
#!/bin/bash
```

```
import csv
a = open('/tmp/test.csv', 'rt')
a_read = csv.reader(a)
print(sum(1 for row in a_read))
```



```
public static void processCsv(Reader iCsv, Writer oCsv, String COL_NAME_SUM) throws IOException {
    CSVPrinter printer = null;
    try {
        printer = new CSVPrinter(oCsv, CSVFormat.DEFAULT.withRecordSeparator(NL));
        List<String> oCsvHeaders = new ArrayList<>();
        CSVParser records = CSVFormat.DEFAULT.withHeader().parse(iCsv);
        Map<String, Integer> iReader = records.getHeaderMap();
        oCsvHeaders = new ArrayList<String>(iReader.keySet().toArray(new String[0]));
        oCsvHeaders.add(COL_NAME_SUM);
        printer.printRecord(oCsvHeaders);
        for (CSVRecord record : records) {
            oCsvRecord = record.getList(record, oCsvHeaders, COL_NAME_SUM);
            printer.printRecord(oCsvRecord);
        }
    } finally {
        if (printer != null) {
            printer.close();
        }
    }
    return;
}
```



DSL for ML Classification

• Concepts

- https://en.wikipedia.org/wiki/Statistical_classification
 - Typically uses CSV file as input eg https://en.wikipedia.org/wiki/Iris_flower_data_set
- Evaluation strategy either:
 - dataset is split in two (training/test), with user defined % training
 - cross-validation (provide means to parameterize it)
- Predictive variables and target variable can be specified
 - By default, all variables are predictive except last column of the CSV (target)
- Specify what to calculate: accuracy and/or recall and/or f1
- Which algorithm(s) to use
 - e.g., classification tree or SVM for scikit-learn

• Services

- Interpreter (ie classification using « random » algorithm)
 - Useful to provide baselines
- Compilers to Python/scikit-learn + (R | Julia)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score
# Using pandas to import the dataset
df = pd.read_csv("iris.csv")
```

```
# Learn more on pandas read_csv :
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html
# pandas input in general :
# https://pandas.pydata.org/pandas-docs/stable/reference/io.html

# Splitting dataset between features (X) and label (y)
X = df.drop(columns=["variety"])
y = df["variety"]

# pandas dataframe operations :
# https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

# Splitting dataset into training set and test set
test_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
```

Example of scikit-learn code to be generated

```
# scikit-learn train_test_split :
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
# Other model selection functions :
# https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection

# Set algorithm to use
clf = tree.DecisionTreeClassifier()

# scikit-learn DecisionTreeClassifier :
# https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
# Other scikit-learn tree algorithms :
# https://scikit-learn.org/stable/modules/classes.html#module-sklearn.tree
# Use the algorithm to create a model with the training set
clf.fit(X_train, y_train)

# Compute and display the accuracy
accuracy = accuracy_score(y_test, clf.predict(X_test))
print(accuracy)
```

DSL for ML Regression

• Concepts

- Evaluation strategy either:
 - dataset is split in two (training/test), with user defined % training
 - cross-validation (provide means to parameterize it)
- Predictive variables and target variable can be specified
 - By default, all variables are predictive except last column of the CSV (target)
- Specify what to calculate: mean relative error...
- Which algorithm(s) to use
 - e.g., regression tree or SVM for scikit-learn

• Services

- Interpreter (ie regression using « random » algorithm)
 - Useful to provide baselines
- Compilers to Python/scikit-learn + (R | Julia)

Tasks

- Choose a sub project among JSON/CSV/ML-classif/ML-reg (Now)
 - Work in groups of up to 2 -> each subproject should be taken at least once
 - Working alone is still possible
 - Insert group composition into spreadsheet by Sept. 24th
 - <https://docs.google.com/spreadsheets/d/1xGhwm5c-4QWzcSAxStwg7hcHPbWRgZhZLiSgcqNZXSs/edit?usp=sharing>
- Build a first version of your metamodel
 - scan of hand written diagram or pdf is good enough at that stage
 - Be ready to present it on Sept. 28th
- Build concrete syntax + parser
 - Be ready to present it on Oct. 5th
- Build interpreter
- Build compiler #1
- Build compiler #2
- Make sure to interoperate with 2 complementary sub-projects
 - Show test case, if ok bonus for all 3 teams.