Detección de errores en entornos de computación distribuida

Inteligencia Artificial
Alberto M. Esmorís Pena

Consultor: David Isern Alarcón
Junio de 2018

Índice

- Objetivos
- Obtención de datos
- Modelo de clasificación
- Software final
- Resultados en entorno de producción
- Conclusiones
- Críticas al trabajo
- Trabajo futuro
- Tecnologías utilizadas

Objetivos

- Detección de errores
  - Errores genéricos
  - Timeouts
- Consultas sencillas sobre los datos
- Software integral funcional

Obtención de datos

- Análisis preliminar : SLURM (*sacct*) y SQOOP (*sqoop-eval*)
- Extracción de datos con Python y SQOOP
- Preparación de datos con Python, Bash y R

Modelo de clasificación

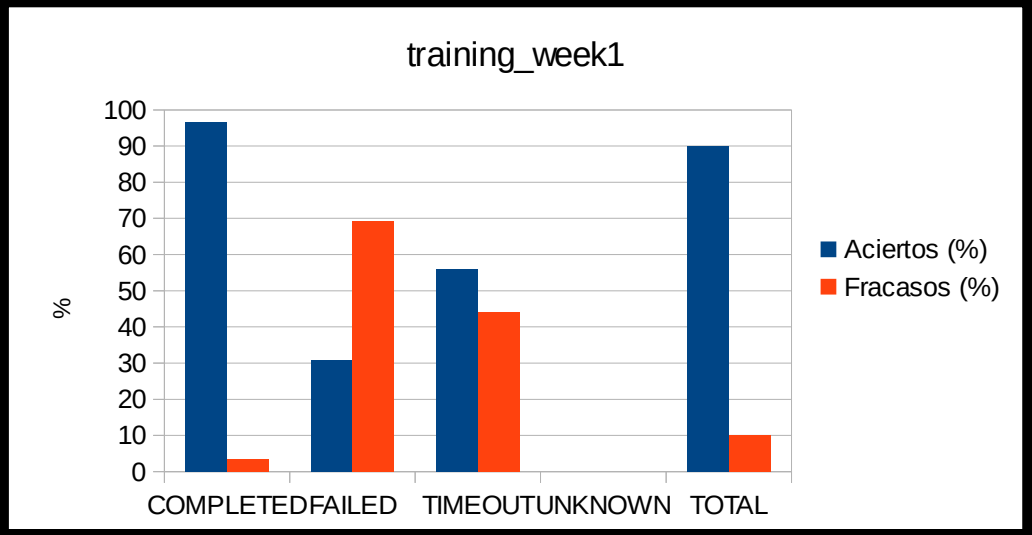
- Script en R
  - Algoritmia C5.0
  - Conjunto de reglas

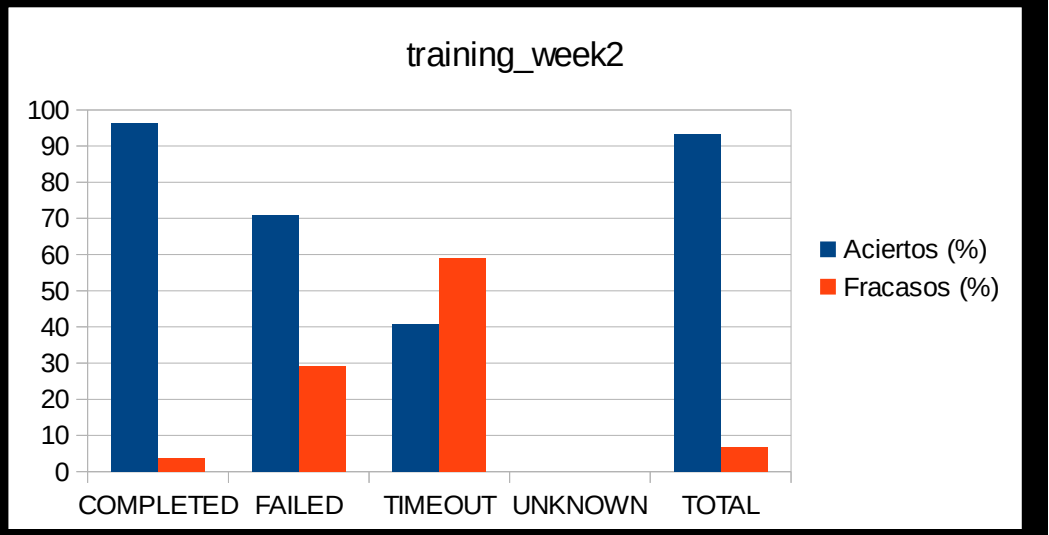
- Experimentos de evaluación

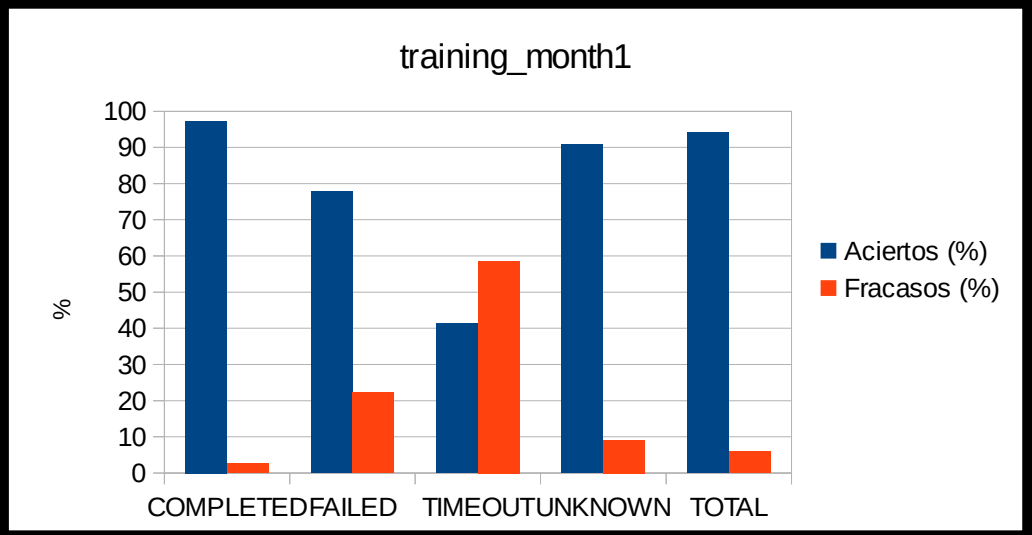
	Training week 1	Training week 2	Training week 3	Taining month 1	Training month 2
Exclusión	*10.75%*	*9.12%*	*11.37%*	*8.63%*	*4.48%*
Error	*10.08%*	*6.77%*	*6.95%*	*5.96%*	*4.25%*

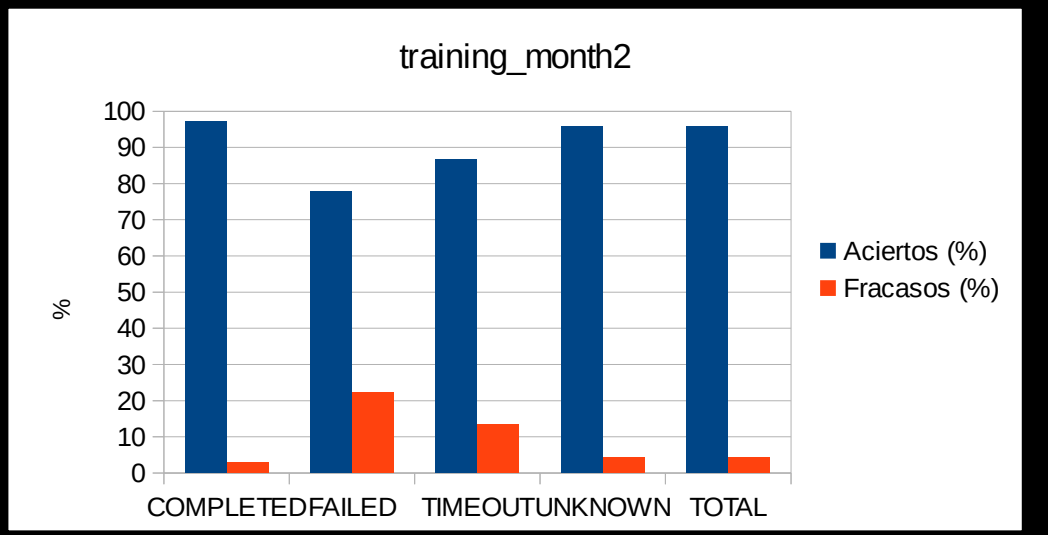
Modelo de clasificación

- Experimentos de evaluación







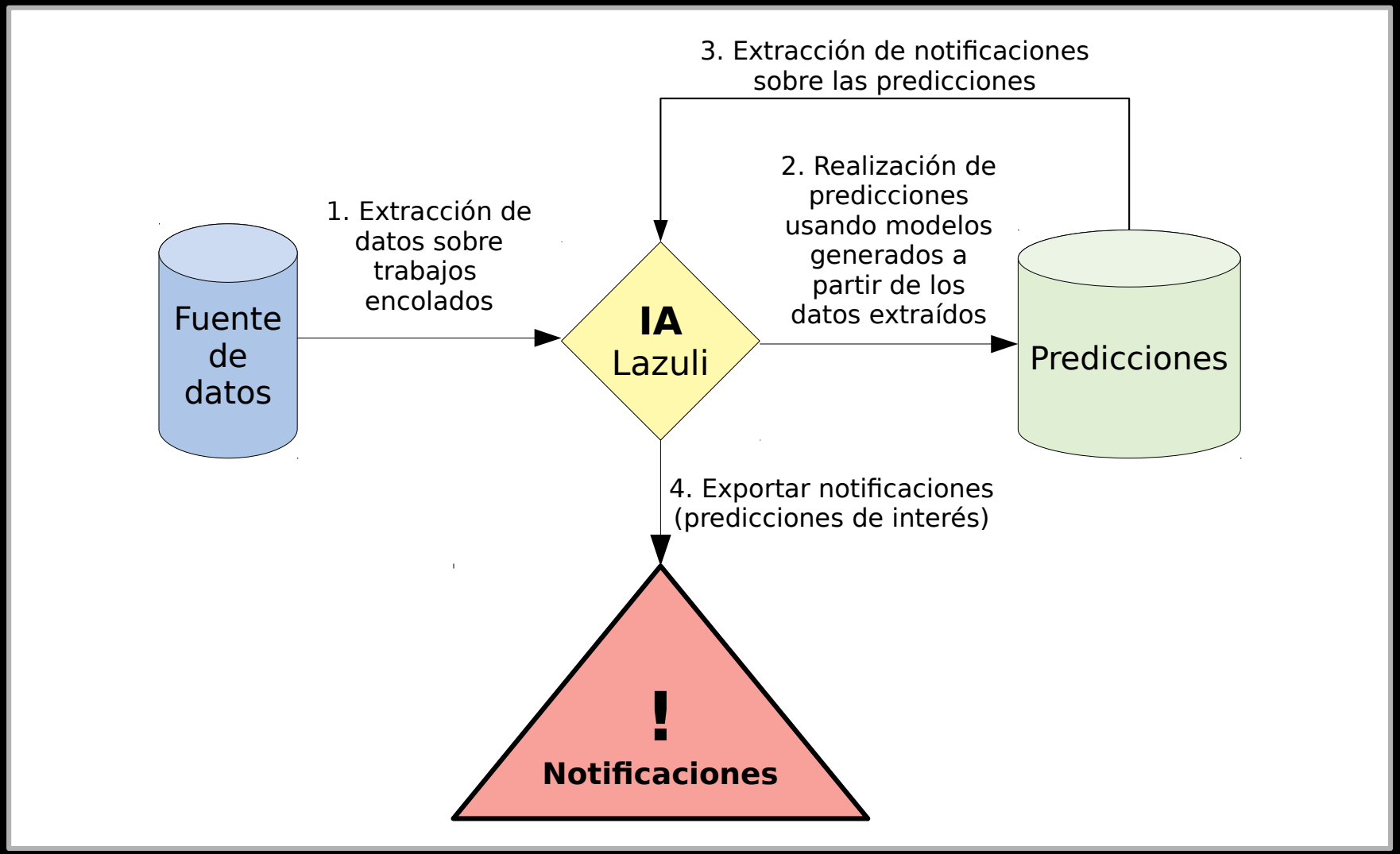


Software final

- Esquema de funcionamiento general
- Código de inicio del servicio de IA
- Archivos de configuración
- Scripts de recolección
- Sistema de purgas

Software final

Esquema de funcionamiento general



Software final

Código de inicio del servicio de IA

Método de ejecución general de la IA

```
public static void exec(String cfgPath) {
        // Obtain config
        Config cfg = Config.getInstance(cfgPath);

        // Configure logger
        Logger.LOGS_DIR = cfg.getLogsPath();
        Logger.println("STARTING AI for jobs classification ...");

        // Build necessary directories
        buildDirs(cfg);

        // Start purger
        startShortDataPurger(cfg);
        if(cfg.isLongDataEnabled())
                startLongDataPurger(cfg);
        startPredictorPredictionsPurger(cfg);
        startPredictorCollectionsPurger(cfg);
        startLogsPurger(cfg);

        // Start collectors
        ModelSynchronizer shortSync = new ModelSynchronizer();
        ModelSynchronizer longSync = new ModelSynchronizer();
        startShortCollector(cfg, shortSync);
        if(cfg.isLongDataEnabled())
                startLongCollector(cfg, longSync);
        startPredictor(cfg, shortSync, longSync);

        Logger.println("AI for jobs classification STARTED");
}
```

Método de inicio de purga de datos

```
public static Purger startShortDataPurger(Config cfg) {
        long ttl = cfg.getShortDataTimeToLive();
        if(ttl <= 0) return null;

        Purger p = new Purger(ttl/3,ttl,
                        new File(cfg.getShortDataPath()));
        p.startPurging();
        return p;
}
```

Software final

Código de inicio del servicio de IA

Método de inicio recolección de datos

```
public static Collector startShortCollector(Config cfg, ModelSynchronizer
shortSync) {
        try {
                Collector c = new Collector(
                        cfg.getShortCollectionInterval(),
                        true,
                        true,
                        cfg.getShortCollectionScript(),
                        cfg.getShortDataPath(),
                        new BuildModelAfterCollection(
                                cfg.getShortModelPath(),
                                new ModelBuilder(
                                        cfg.getShortModelScript(),
                                        shortSync
                                )
                        ));
                c.start();
                return c;
        }catch(Exception ex) {
                String err = "ERROR starting short collector";
                Logger.println(err);
                Logger.printException(ex);
                System.exit(1);
        }
        return null; // Should NEVER be reached because of System.exit
inside catch
}
```

Método de inicio de predicciones

```
public static Collector startPredictor(Config cfg, ModelSynchronizer shortSync,
ModelSynchronizer longSync) {
        try {
                Collector p = new Collector(cfg.getPredictorInterval(),
                        false,
                        true,
                        cfg.getPredictorScriptPath(),
                        cfg.getPredictorCollectionPath(),
                        new PredictAfterCollection(
                                cfg.getShortModelPath(),
                                cfg.getLongModelPath(),
                                shortSync,
                                longSync,
                                cfg.getPredictorPredictionsPath(),
                                null,
                                cfg)
                        );
                ((PredictAfterCollection)p.getAfterCollectionStrategy()).setPredictor(p);
                String oldCmd[] = p.getCollectionCommand();
                String newCmd[] = Arrays.copyOf(oldCmd, oldCmd.length+3);
                newCmd[newCmd.length-3] = cfg.getPredictorLastCollectionEpoch()+""; // $2
                newCmd[newCmd.length-2] = (System.currentTimeMillis()/1000)+""; // $3
                newCmd[newCmd.length-1] = cfg.getNotifierPool(); // $4
                cfg.setPredictorLastCollectionEpoch(
                        Long.parseLong(newCmd[newCmd.length-2]));
                cfg.save();
                p.setCollectionCommand(newCmd);
                p.start();
                return p;
        }catch(Exception ex) {
                String err = "ERROR starting predictor";
                Logger.println(err);
                Logger.printException(ex);
                System.exit(1);
        }
        return null; // Should NEVER be reached because of System.exit inside catch
}
```

Software final

Archivo de configuración general

```
PredictorCollectionsTimeToLive=300
PredictorInterval=3600
ShortModelScript=data/c50.R
LongDataPath=data/long/collections
LongDataEnabled=true
ShortCollectionInterval=2592000
ShortCollectionScript=data/short/short_collection_script.sh
ShortModelPath=data/short/short.rules
LongCollectionInterval=7776000
PredictorPath=data/predictor/
LongCollectionScript=data/long/long_collection_script.sh
NotifierConfiguration=data/notifier/config.cfg
LongDataTimeToLive=10368000
ShortDataPath=data/short/collections
PredictorScriptPath=data/predictor/data_to_predict_collection_script.sh
NotificationsPath=data/notifier/notifications.csv
PredictorLastCollectionEpoch=1524650420
LongModelPath=data/long/long.rules
LogsPath=logs
LongModelScript=data/c50.R
PredictorPredictionsPath=data/predictor/predictions
ShortDataTimeToLive=5184000
PredictorPredictionsTimeToLive=7200
LogsTimeToLive=15552000
PredictorCollectionPath=data/predictor/collections
NotifierPool=data/notifier/pool
```

Archivo de configuración de notificaciones

```
StateFilter=[FAILED,TIMEOUT]
MinMemReq=null
StartWeekDayFilter=null
MaxNodesAlloc=null
MinNodesAlloc=null
MinStartTime=null
MinCpusAlloc=null
MaxMemReq=null
MinCpusReq=null
MinAccuracy=0.90
MaxCpusAlloc=null
UserFilter=null
NodeFilter=null
MaxCpusReq=null
MinElapsedTime=null
AccountFilter=null
PartitionFilter=null
MaxStartTime=null
MaxElapsedTime=null
```

Software final

Script de recolección de datos para la construcción del modelo principal

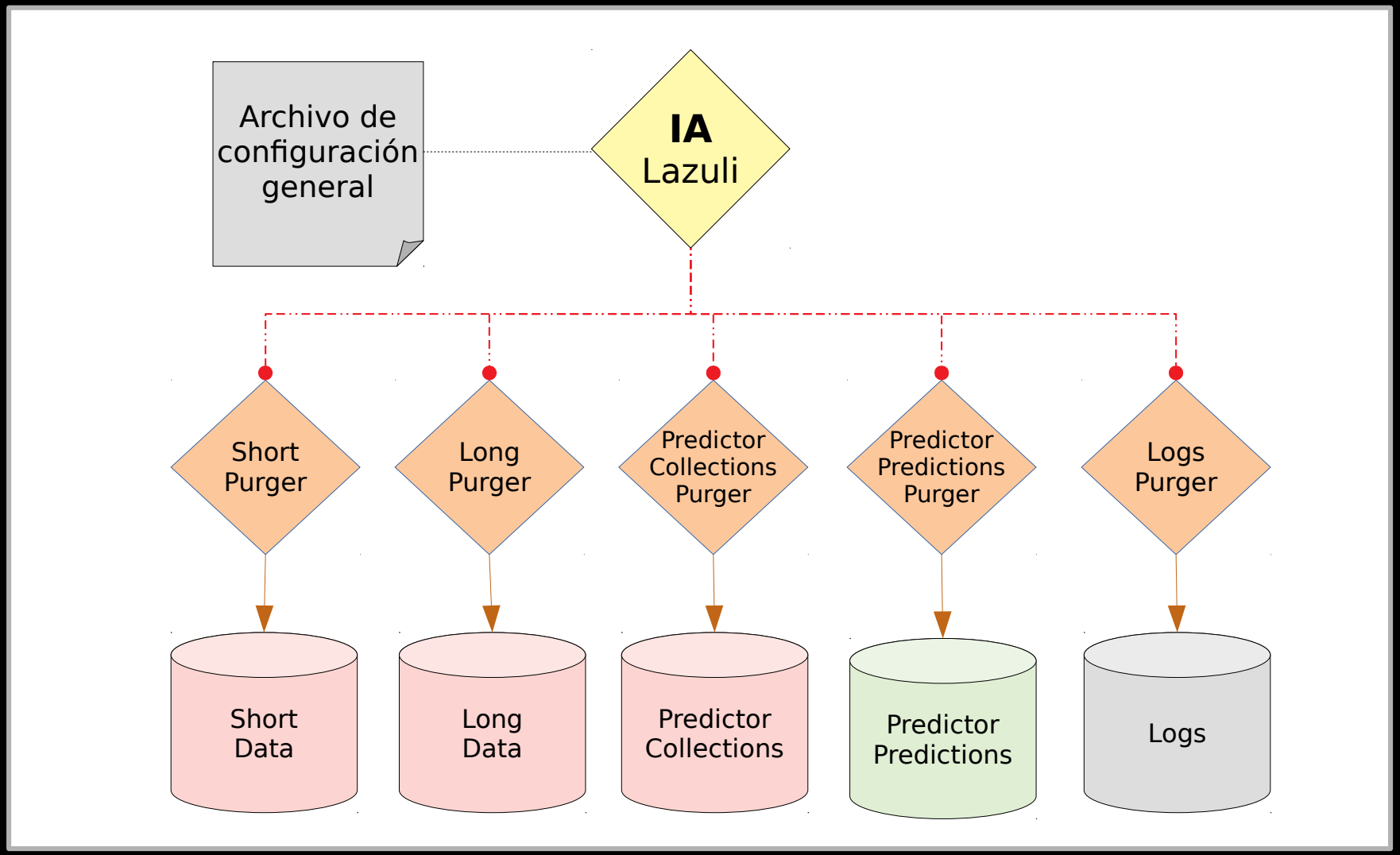
```
#!/bin/bash
# $1 -> Directory where the file will be exported

if [ $# -lt 1 ]; then
cat << EOF
No directory was specified.
EOF
      exit 1
fi


now=`date +%s`

start_epoch=$(($now-2592000))
end_epoch=$now

start_date=`date -d @$start_epoch +"%Y-%m-%dT%H:%M:%S"`
end_date=`date -d @$end_epoch +"%Y-%m-%dT%H:%M:%S"`

file_name='SD_FROM_'$start_date'_TO_'$end_date'.csv'

python export_metrics/export_metrics.py -out $1/$file_name
      -started-after-ts $start_epoch -started-before-ts $end_epoch
data/fix.sh $1/$file_name $1/$file_name'.tmp'
grep -v RUNNING $1/$file_name'.tmp' > $1/$file_name

echo $1/$file_name
```

Script de recolección de datos a clasificar

```
#!/bin/bash
# $1 -> Directory where the file will be exported
# $2 -> Collection start epoch
# $3 -> Collection end epoch
# $4 -> Path to pool of notifications that must be included

if [ $# -lt 1 ]; then
cat << EOF
No directory was specified.
EOF
      exit 1
fi


now=`date +%s`

start_epoch=$2
end_epoch=$3

start_date=`date -d @$start_epoch +"%Y-%m-%dT%H:%M:%S"`
end_date=`date -d @$end_epoch +"%Y-%m-%dT%H:%M:%S"`

file_name='DATA_FROM_'$start_date'_TO_'$end_date'.csv'

if [ -f $4 ]; then
      python export_metrics/export_metrics.py -out $1/$file_name
            -started-after-ts $start_epoch -started-before-ts
            $end_epoch -running-only -extra-queries $4
else
      python export_metrics/export_metrics.py -out $1/$file_name
            -started-after-ts $start_epoch -started-before-ts
            $end_epoch -running-only
fi
data/fix.sh $1/$file_name $1/$file_name'.tmp'
mv $1/$file_name'.tmp' $1/$file_name

echo $1/$file_name
```

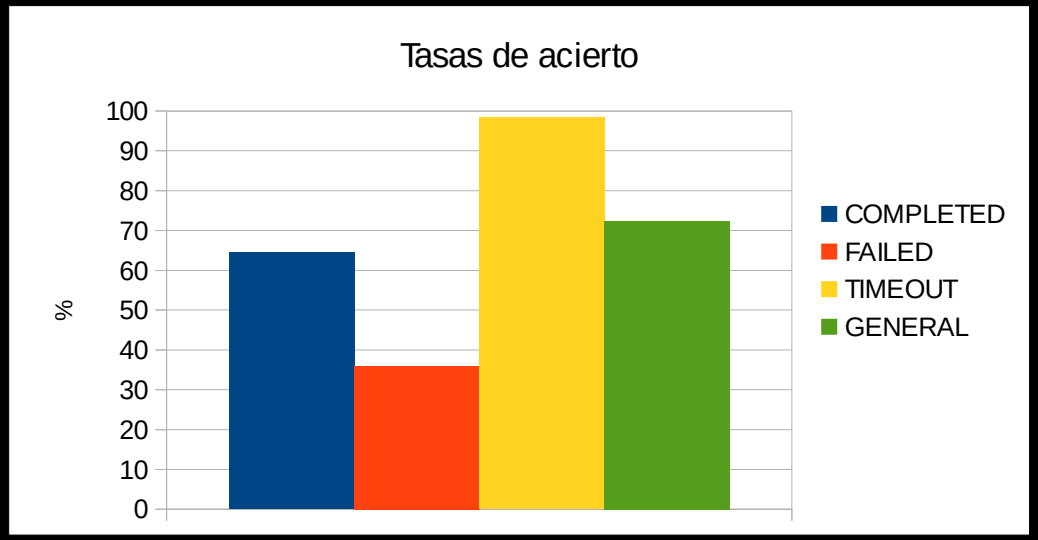
Software final

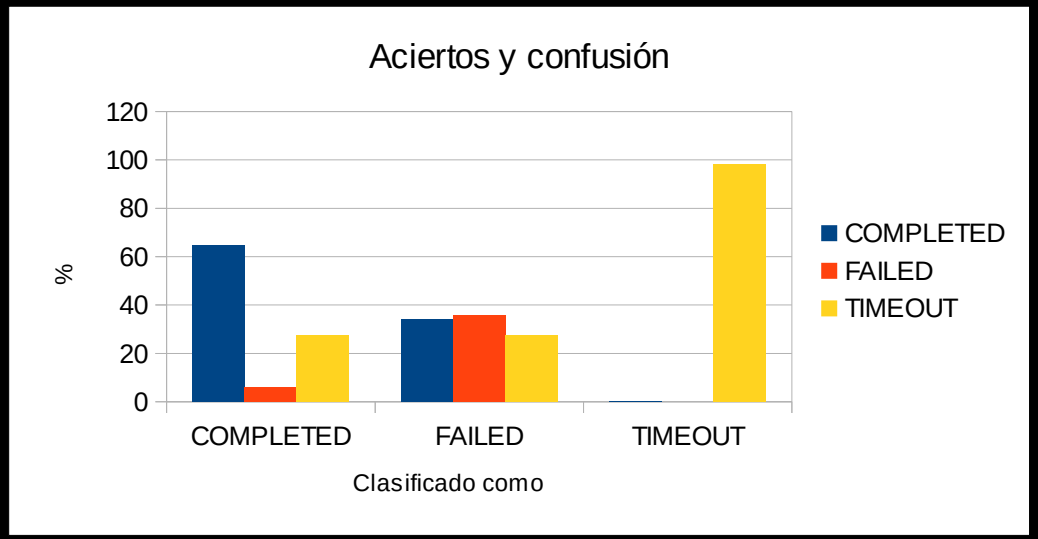
Esquema del sistema de purgas

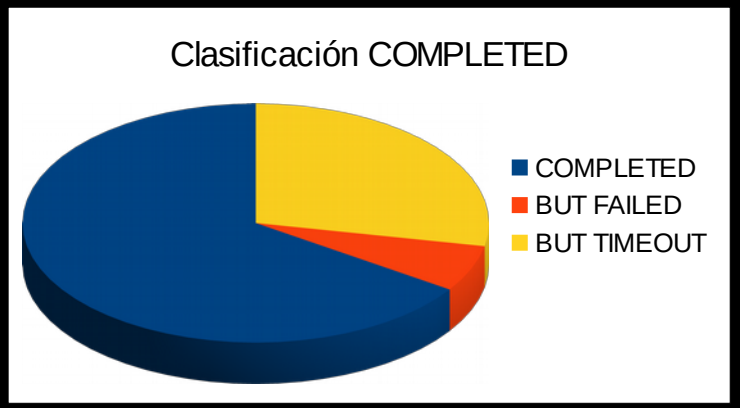


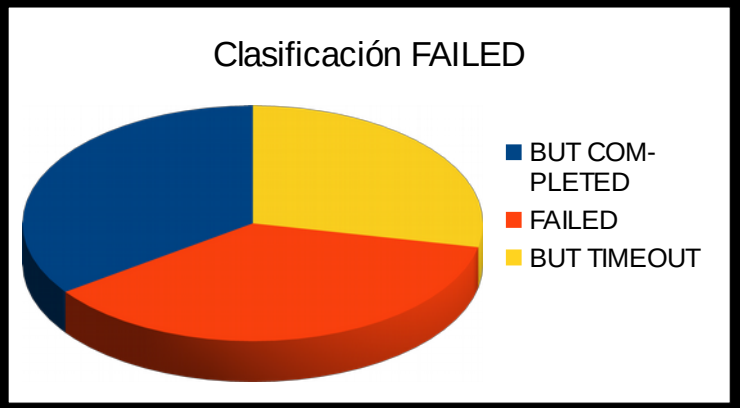
Resultados en entorno de producción

- Análisis de predicciones



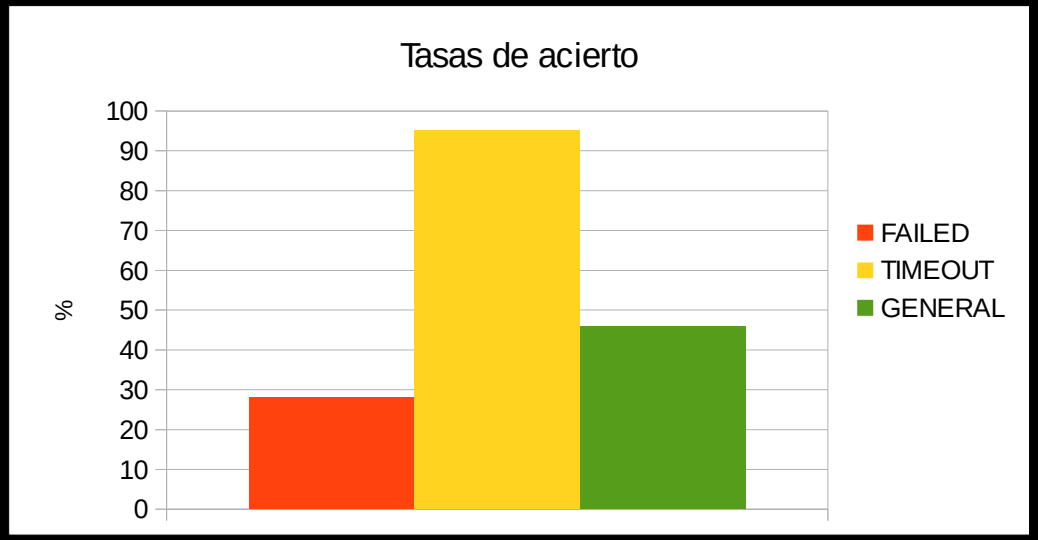


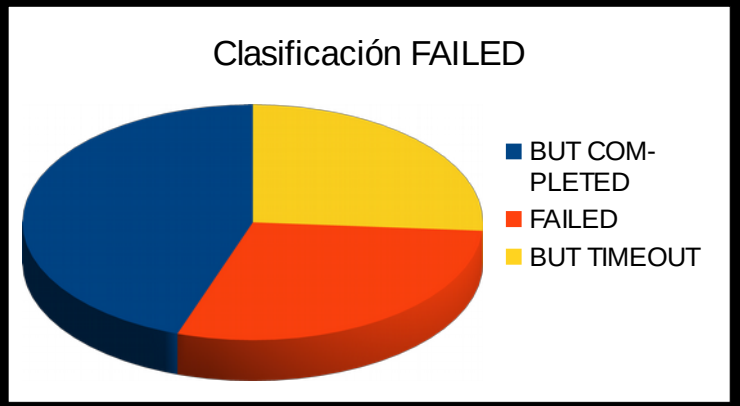


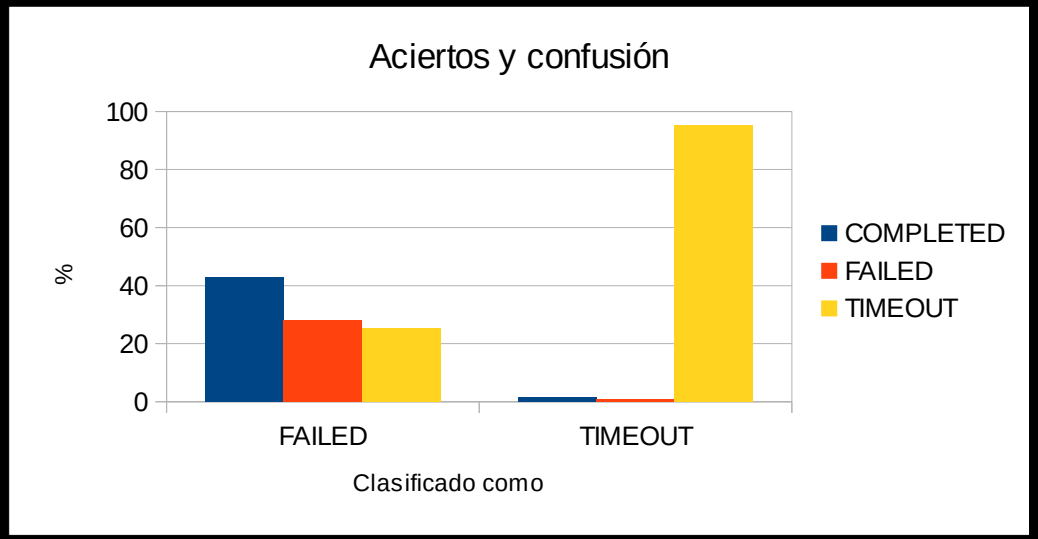


Resultados en entorno de producción

- Análisis de notificaciones







Conclusiones

- Uso de diferentes herramientas
- Integración de distintos componentes como producto final
- Entorno real frente a entorno académico

Críticas al trabajo

- Objetivos logrados
  - Clasificaciones precisas en entorno NO real
  - Detección de timeouts en entorno real
  - Sistema de consultas/filtros
- Precisión variable
- El atributo *time_elapsed*

Trabajo futuro

- Integración con otras técnicas de Inteligencia Artificial
  - Detección de anomalías en series temporales
  - Uso inteligente de distintos modelos de clasificación
- Interfaz gráfica
  - Para gestión del servicio (poco importante)
  - Para visualización de notificaciones
  - Para el sistema de consultas
- Aplicación en otros ámbitos

Tecnologías utilizadas

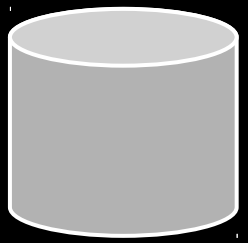
Python

Java

R

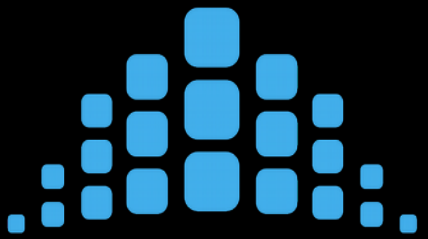
Bash

Anaconda



SQL

SQOOP



SLURM

Gracias por su tiempo