

**PROJECT
REPORT:
Automating
Workflows in
KNIME**

Naveed Hassan

Table of Contents

Objective and Approach	3
Process	3
Designing Workflows	3
Batch File and Commands.....	4
Scheduling for Automation.....	5
Windows Task Scheduler Method	5
Node-RED Method	5
Designing a Scheduler using Python.....	6
Conclusion.....	6
References.....	6

Objective and Approach

The objective of this project is to Automate workflows in KNIME without accessing the premium (paid) KNIME server. The approach taken to try and achieve this was by executing the workflow through the command line. This was done by running a batch file. Now, to automate this, the methods possible identified were by using scheduling. There was multiple possibilities to implement this: Windows Task Scheduler, Node-RED or building a low-code scheduler from scratch on Python. All 3 of the methods were implemented and tested out and are illustrated below.

Process

Designing Workflows

A simple workflow was designed on KNIME Analytics Platform to figure out how a workflow runs on KNIME and to test out the automation of workflows. Below is an illustration of the workflow built.

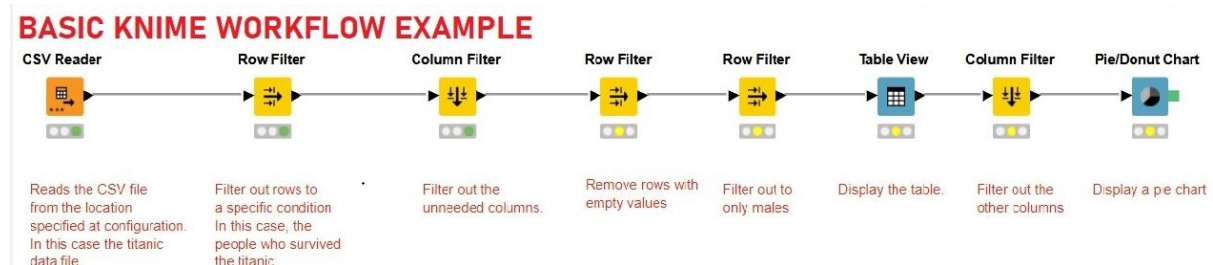


Image 1: Workflow Diagram

This is a simple workflow which reads a csv file of data and filters out certain rows and columns and finally outputs a pie chart. This is a very basic example of a workflow in KNIME and was used to test the automation of workflows. The workflows can be edited to filter the data and parameters according to the query.

A simple Machine Learning Model was also built into a workflow on KNIME.

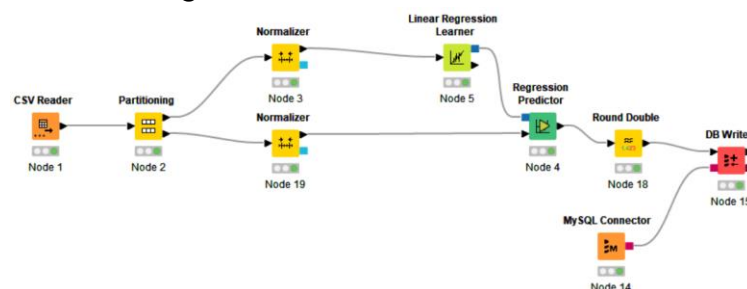


Image 2: Machine Learning Model Workflow Diagram

The workflow builds a model to predict a score given a set of data and output the predicted data into a database.

Batch File and Commands

A simple command was put into the batch file as shown below^[1]

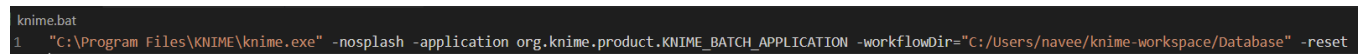
```
"C:\Program Files\KNIME\knime.exe" -nosplash -application  
org.knime.product.KNIME_BATCH_APPLICATION  
-workflowDir="C:/Users/<user>/knime-workspace/WF" -reset
```

Arguments:

- "C:\Program Files\KNIME\knime.exe"**: Launches KNIME.
- consoleLog**: Causes a new window to be opened containing the log messages and will keep the window open after the execution has finished.
- reset**: Reset workflow prior to execution.
- workflowDir="C:/Users/<user>/knime-workspace/workflow"**: Shows the directory of the workflow to be executed.

We can edit the command line to switch between different workflows when required.

Enter the command into a file and save as a .bat file.



```
knime.bat  
1 "C:\Program Files\KNIME\knime.exe" -nosplash -application org.knime.product.KNIME_BATCH_APPLICATION -workflowDir="C:/Users/navee/knime-workspace/Database" -reset
```

Image 3: Batch File

When this file is executed, it will run the workflow and output a return code to show Successful/Unsuccessful execution in a prompt which will read as follows^[2]:

Return Code = 0 upon successful execution

Return Code = 2 if parameters are wrong or missing

Return Code = 3 when an error occurs during loading a workflow

Return Code = 4 if an error during execution occurred

Scheduling for Automation

Windows Task Scheduler Method

This method uses the built in Windows Task Scheduler to schedule the computer to run the batch file after a specific interval for a certain amount of time. This allows the program to run in the background and can even wake the computer to begin the task.

Node-RED Method

Node-RED allows us to build flows to inject & repeat at intervals to replicate automation. Although, one downside is that Node-RED must be open at all times for the flow to be running and injecting.

Node-RED FLOW

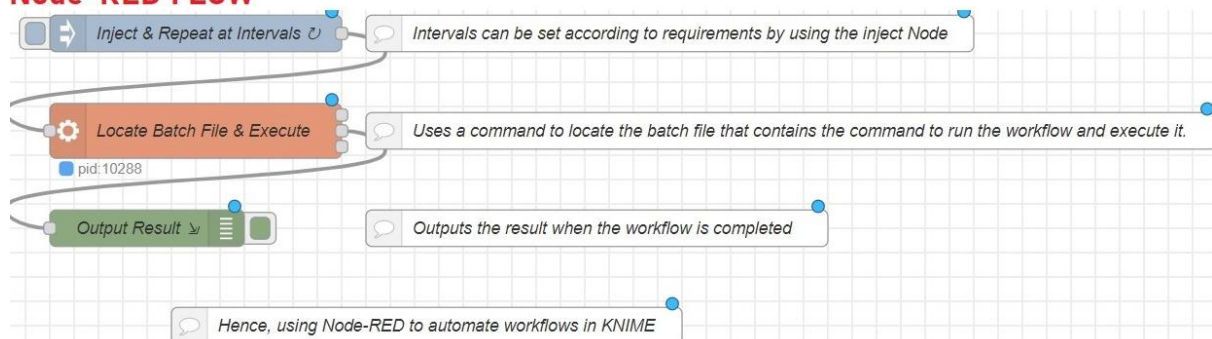


Image 4: Node-RED Flow

A simple flow was built to execute the batch file mentioned above to eventually automate running workflows in KNIME.

```
INFO: Removing the extensions for bundle 427
Nov 23, 2022 2:26:05 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener unregister
INFO: Removing the extensions for bundle 428
23 Nov 14:26:14 - [info] [debug:Output Result]
Nov 23, 2022 2:26:05 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener addExtensions
INFO: Adding the extensions from bundle org.apache.cxf.cxf-rt-frontend-jaxrs (423) [org.apache.cxf.jaxrs.JAXRSBindingFactory]
Nov 23, 2022 2:26:05 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener addExtensions
INFO: Adding the extensions from bundle org.apache.cxf.cxf-rt-transports-http (427) [org.apache.cxf.transport.http.HTTPTransportFactory, org.apache.cxf.transport.http.HTTPWSSEExtensionLoader, org.apache.cxf.transport.http.policy.HTTPClientAssertionBuilder, org.apache.cxf.transport.http.policy.HTTPServerAssertionBuilder, org.apache.cxf.transport.http.policy.NoOpPolicyInterceptorProvider]
Nov 23, 2022 2:26:05 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener addExtensions
INFO: Adding the extensions from bundle org.apache.cxf.cxf-rt-transports-http-hc (428) [org.apache.cxf.transport.http.HTTPConduitFactory, org.apache.cxf.transport.ConduitInitiator]
Nov 23, 2022 2:26:05 PM org.apache.cxf.blueprint.NamespaceHandlerRegisterer register
INFO: Aries Blueprint packages not available. So namespaces will not be registered
Nov 23, 2022 2:26:05 PM org.apache.cxf.blueprint.NamespaceHandlerRegisterer register
INFO: Aries Blueprint packages not available. So namespaces will not be registered
INFO: Removing the extensions for bundle 423
Nov 23, 2022 2:26:14 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener unregister
INFO: Removing the extensions for bundle 427
Nov 23, 2022 2:26:14 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener unregister
INFO: Removing the extensions for bundle 428
23 Nov 14:31:14 - [info] [debug:Output Result]
Nov 23, 2022 2:31:05 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener addExtensions
INFO: Adding the extensions from bundle org.apache.cxf.cxf-rt-frontend-jaxrs (423) [org.apache.cxf.jaxrs.JAXRSBindingFactory]
Nov 23, 2022 2:31:05 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener addExtensions
INFO: Adding the extensions from bundle org.apache.cxf.cxf-rt-transports-http (427) [org.apache.cxf.transport.http.HTTPTransportFactory, org.apache.cxf.transport.http.HTTPWSSEExtensionLoader, org.apache.cxf.transport.http.policy.HTTPClientAssertionBuilder, org.apache.cxf.transport.http.policy.HTTPServerAssertionBuilder, org.apache.cxf.transport.http.policy.NoOpPolicyInterceptorProvider]
Nov 23, 2022 2:31:05 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener addExtensions
INFO: Adding the extensions from bundle org.apache.cxf.cxf-rt-transports-http-hc (428) [org.apache.cxf.transport.http.HTTPConduitFactory, org.apache.cxf.transport.ConduitInitiator]
Nov 23, 2022 2:31:14 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener unregister
INFO: Removing the extensions for bundle 423
Nov 23, 2022 2:31:14 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener unregister
INFO: Removing the extensions for bundle 427
Nov 23, 2022 2:31:14 PM org.apache.cxf.bus.osgi.CXFExtensionBundleListener unregister
INFO: Removing the extensions for bundle 428
```

Image 5: Workflow execution

The image above shows proof that the workflow is being executed correctly at the correct intervals.

Designing a Scheduler using Python

A low-code simple scheduler was built from scratch using python to clone the process of automation. Again, the batch file was used in order to execute the workflow.

```
def knime() -> str:
    """
    Function to run the batch file containing the command to run the workflow in KNIME
    :Output:
    | Returns the outcome of the execution (Successful or Unsuccessful)
    """
    # Execute batch file
    os.system("C:\\Users\\navee\\OneDrive\\Desktop\\Knime\\knime.bat")

    # Check return code
    check = os.system("C:\\Users\\navee\\OneDrive\\Desktop\\Knime\\knime.bat")

    # Get current date and time
    t = dt.datetime.now()

    # Reformat Date and Time
    t = t.strftime('%d/%m/%Y %H:%M:%S')

    # Return Code = 0 upon successful execution
    if check == 0:
        return str("Successfully Executed at " + t)

    # Return Code = 2 if parameters are wrong or missing
    elif check == 2:
        return str("Incorrect or Missing Parameters. Time: " + t)

    # Return Code = 3 when an error occurs during loading a workflow
    elif check == 3:
        return str("An error occurred when loading a workflow. Time: " + t)

    # Return Code = 4 if an error during execution occurred
    elif check == 4:
        return str("an error occurred during execution. Time: " + t)
```

Image 6: Snippet of code to execute the workflow

```
def schedule(interval: int, duration: int) -> None:
    """
    Function to schedule to run the knime function at specific intervals
    :Input:
    | interval: int: of the amount of time between execution (given in minutes)
    | duration: int: Amount of time to run this interval for (Hours)
    """
    # Convert interval to seconds
    interval = interval * 60 - 30

    # Convert duration to seconds
    duration = duration * 60 * 60

    end_time = time.time() + duration
    while time.time() < end_time:
        # Run the knime workflow function
        print(knime())

        # Repeat after specific interval
        time.sleep(interval)
```

Image 7: Snippet of code to replicate scheduling

The above was used to code a program that acts as a scheduler on Python to run the KNIME workflow after a certain interval for a certain amount of time.

Conclusion

To conclude, there exists multiple methods of achieving the goal of automate the KNIME workflows without the use of the KNIME server. The infrastructure of the process is based on the command line in the batch file which is used in every method. The automation part has multiple options in terms of scheduling including building a scheduler from scratch.

References

- [1] KNIME FAQ. Available at: <https://www.knime.com/faq#q12>
- [2] KNIME (2019) *Execute workflow in batch mode Windows 10*, KNIME Community Forum. Available at: <https://forum.knime.com/t/execute-workflow-in-batch-mode-windows-10/13986>