

**Lab Guide**

**Machine Learning with  
Pentaho Data Integration (PDI)**

**Credit Card Fraud – randomForest in R**

# HITACHI

## Inspire the Next

Change log:

Date	Courseware Version	Microcode	Notes
June 2020	1.0	9.0.0.3 - 582	

# Contents

Terminology.....	1
Prerequisites.....	2
Installing Python .....	2
Installing Libraries.....	4
Google Colab.....	5
Installing R.....	6
Setting R Environment Variables .....	7
Installing the randomForest Library.....	11
Installing RStudio (Optional).....	12
Configuring Pentaho Data Integration with R.....	13
Verifying Your Installation .....	13
Lab 1: Credit Card Fraud – AutoML .....	15
Objectives .....	15
Step 1 – Data Preparation.....	15
Step 2 – Feature Engineering.....	18
Step 3 – Test Machine Learning Models to Identify the Most Accurate Model.....	18
TPOT: py-auto_ml.....	18
Colab.....	20
AutoML Script.....	21
Python Executor.....	23
Lab 2: Credit Card Fraud – randomForest .....	29
Objectives .....	29
Train the Model.....	29
R Script Executor: Train Model .....	30
Predict Fraudulent Credit Card Transactions .....	33
Appendix A: Python Script for PDI.....	37
Appendix C: R Script for Train.....	41
Appendix D: RScript for Predict .....	42
Related Information .....	43



# Terminology

The table below outlines the different course activities:

Activity	Description
<b>Demonstration</b>	The Instructor will demonstrate the workflow, outlining the key concept(s). The student is not expected to replicate the Instructor's demonstration; but understand the key concept(s) and workflow.
<b>Lab</b>	The Instructor will outline the key concepts, features and options. The student is expected to follow along with the Instructor so that they understand the key concept(s), features and options for the Exercise.



*The icon indicates an Info Tip. Info Tips help users understand unfamiliar workflows or actions.*



*The icon indicates that you need to be careful when implementing or configuring the step/option(s).*



*The icon indicates a Best Practice. A Best Practice is a method or technique that has been generally accepted as a standard way of doing things.*

## Prerequisites

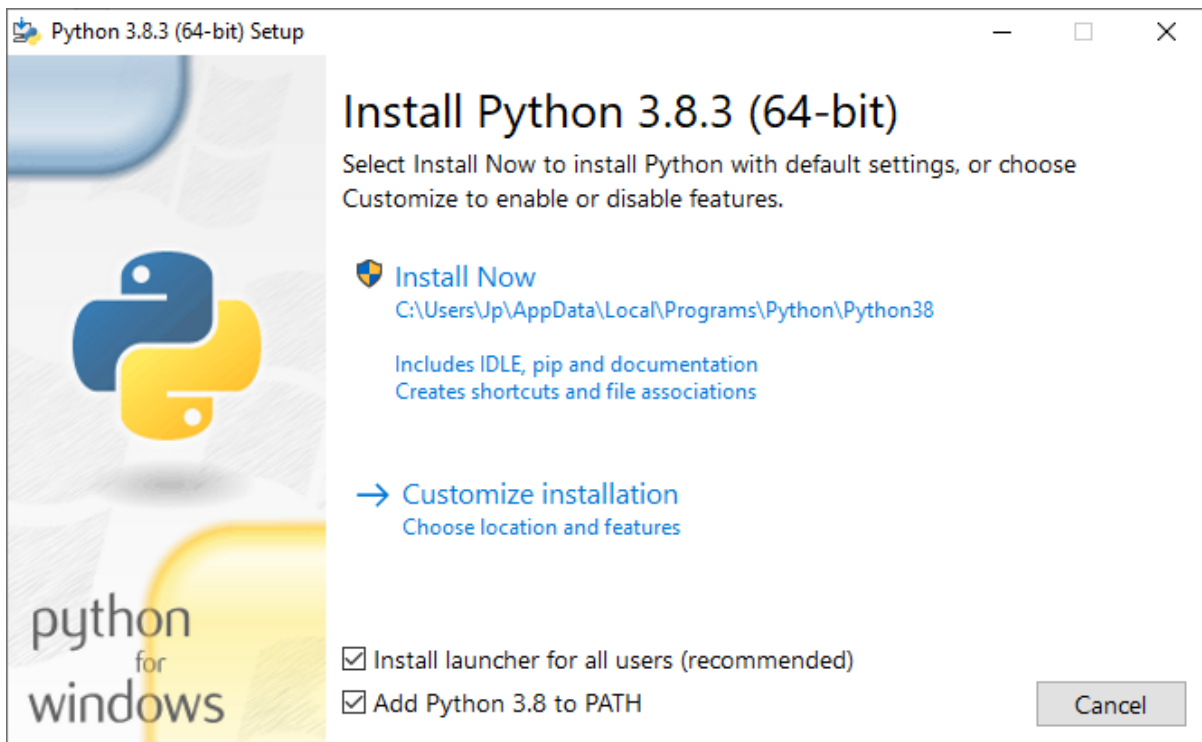
The following prerequisites need to be completed:

- Install Python for Windows
- Google Colab
- Install R for Windows
- Set R Environmental Variables
- Install R Studio for Windows
- Configure Pentaho Data Integration with R

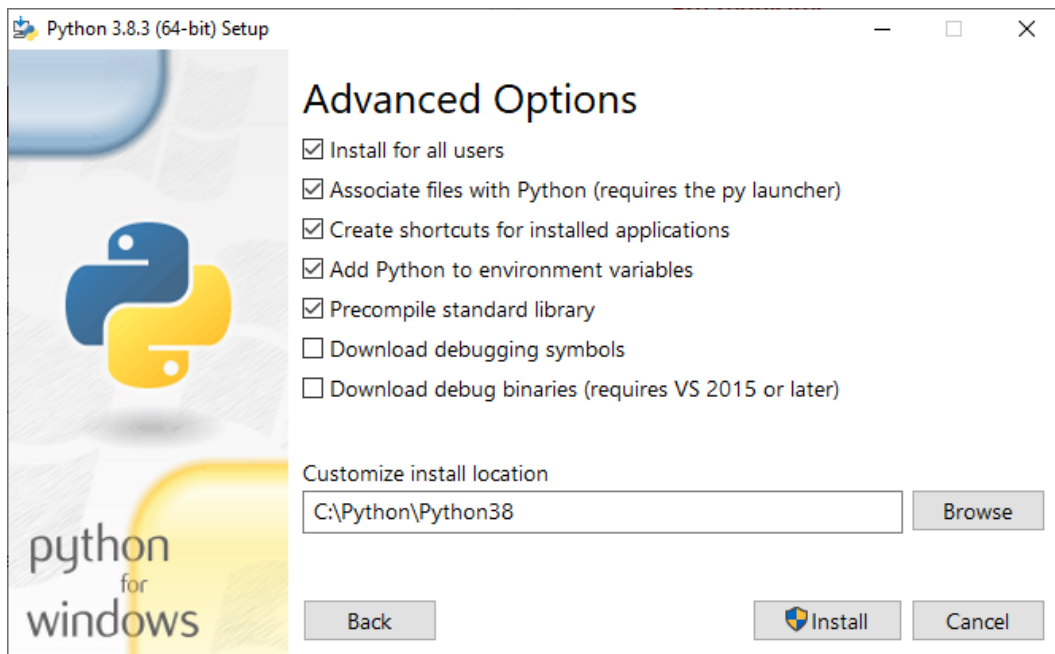
## Installing Python

To install Python:

1. Download Python from [Python for Windows](#).
2. Click on **Download Windows x86-64 executable installer** on the page.
3. Run `python-3.8.3-amd64.exe` and follow the installation instructions.



4. Ensure you select:
  - a. Customize Installation
  - b. Install launcher
  - c. Add Python 3.8 to PATH
5. Keep default options and click **Next**.
6. Ensure you select:
  - a. Install for all users
  - b. Precompile standard library
  - c. Change the Path to: C : \Python\Python38



7. Click **Install**.

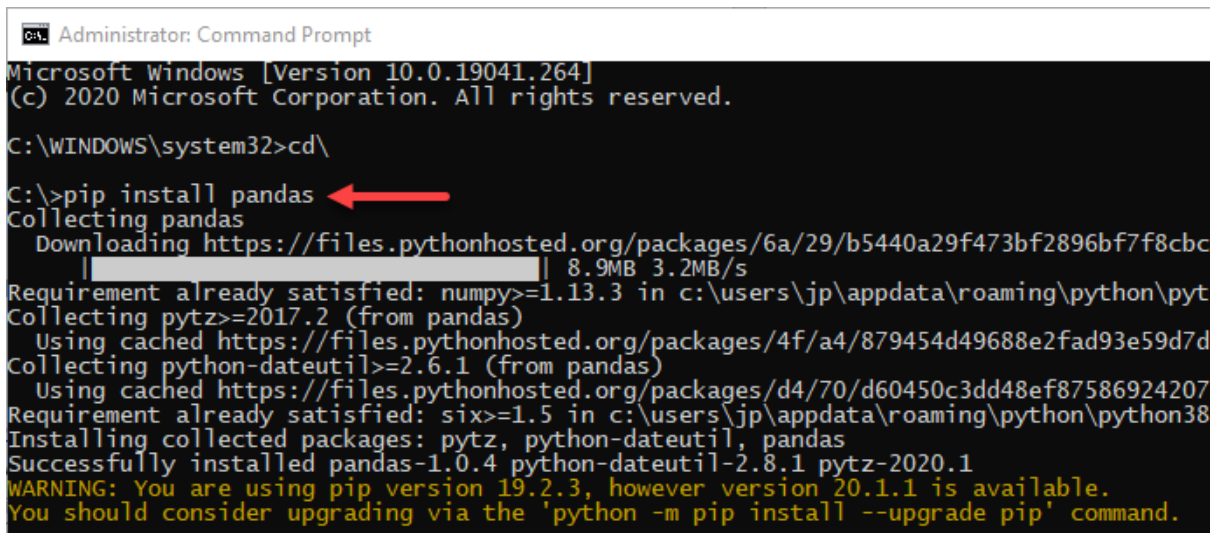
## Installing Libraries

The following libraries need to be installed:

- pandas
- matplotlib
- py4j
- numpy
- wheel
- sklearn
- TPOT

To do this:

1. Open a Command Prompt (Admin) window.
2. Enter the following command to install pandas: `pip install pandas`



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd\

C:\>pip install pandas
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/6a/29/b5440a29f473bf2896bf7f8cbc
  [REDACTED] 8.9MB 3.2MB/s
Requirement already satisfied: numpy>=1.13.3 in c:\users\jp\AppData\Roaming\Python\Python38\
Collecting pytz>=2017.2 (from pandas)
  Using cached https://files.pythonhosted.org/packages/4f/a4/879454d49688e2fad93e59d7d
Collecting python-dateutil>=2.6.1 (from pandas)
  Using cached https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586924207
Requirement already satisfied: six>=1.5 in c:\users\jp\AppData\Roaming\Python\Python38
Installing collected packages: pytz, python-dateutil, pandas
Successfully installed pandas-1.0.4 python-dateutil-2.8.1 pytz-2020.1
WARNING: You are using pip version 19.2.3, however version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

3. Repeat to download and install the other required libraries:



```
Administrator: Command Prompt
Found existing installation: pip 19.2.3
Uninstalling pip-19.2.3:
Successfully uninstalled pip-19.2.3
Successfully installed pip-20.1.1

C:\>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.2.1-cp38-cp38-win_amd64.whl (9.2 MB)
    | 9.2 MB 3.3 MB/s
Requirement already satisfied: numpy>=1.11 in c:\users\jp\AppData\Roaming\Python\Python38\site-packages (from matplotlib)
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1
  Using cached pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.2.0-cp38-none-win_amd64.whl (58 kB)
    | 58 kB 4.1 MB/s
Collecting cyclar>=0.10
  Using cached cyclar-0.10.0-py2.py3-none-any.whl (6.5 kB)
Requirement already satisfied: python-dateutil>=2.1 in c:\python38\lib\site-packages (from cyclar>=0.10)
Requirement already satisfied: six in c:\users\jp\AppData\Roaming\Python\Python38\site-packages (from python-dateutil>=2.1)
Installing collected packages: pyparsing, kiwisolver, cyclar, matplotlib
Successfully installed cyclar-0.10.0 kiwisolver-1.2.0 matplotlib-3.2.1 pyparsing-2.4.7

C:\>
```

You may need to restart your system.

## Google Colab

Colab is a Python development environment, based on Jupyter Notebooks, that runs in the browser using Google Cloud. It provides a runtime, fully configured for deep learning libraries, such as Keras, TensorFlow, PyTorch, and OpenCV.

- For further details: [Google Colab](#)
- Recommended to sync with Google Drive.

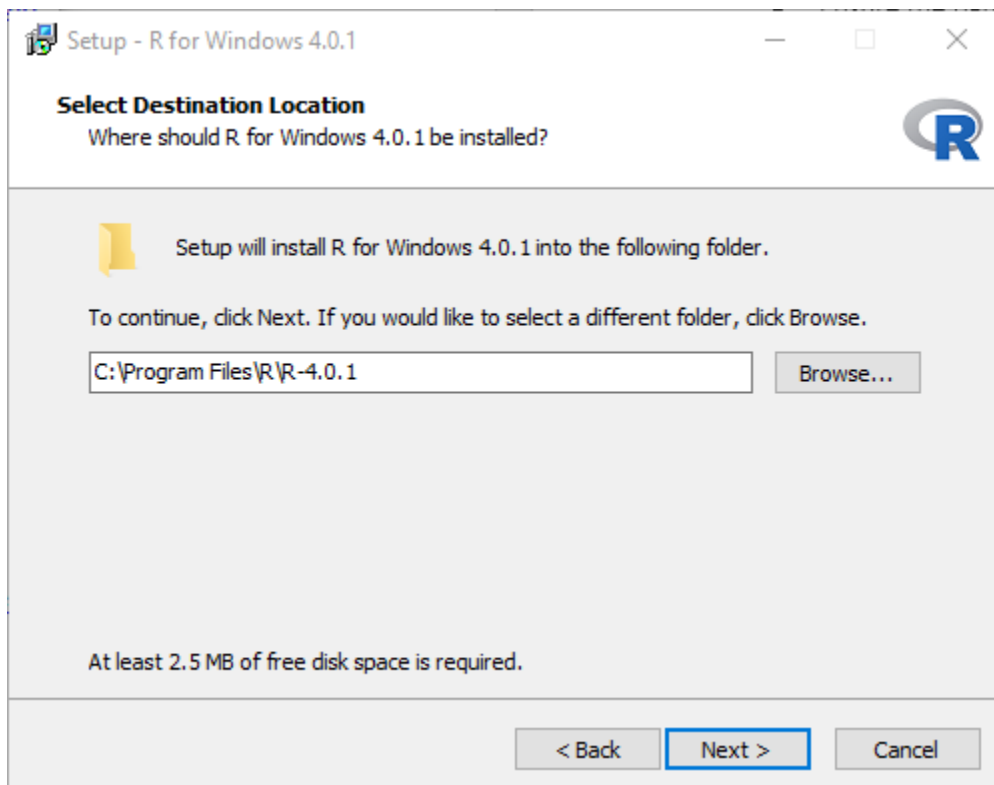
## Installing R

To install R:

1. Download R from [r-project](https://www.r-project.org/) by selecting a CRAN location (a mirror site) and clicking the corresponding link.
2. Click **Download R for Windows** at the top of the page.
3. Click **install R for the first time** at the top of the page.

[Download R 4.0.1 for Windows](#) (84 megabytes, 32/64 bit)  
[Installation and other instructions](#)  
[New features in this version](#)

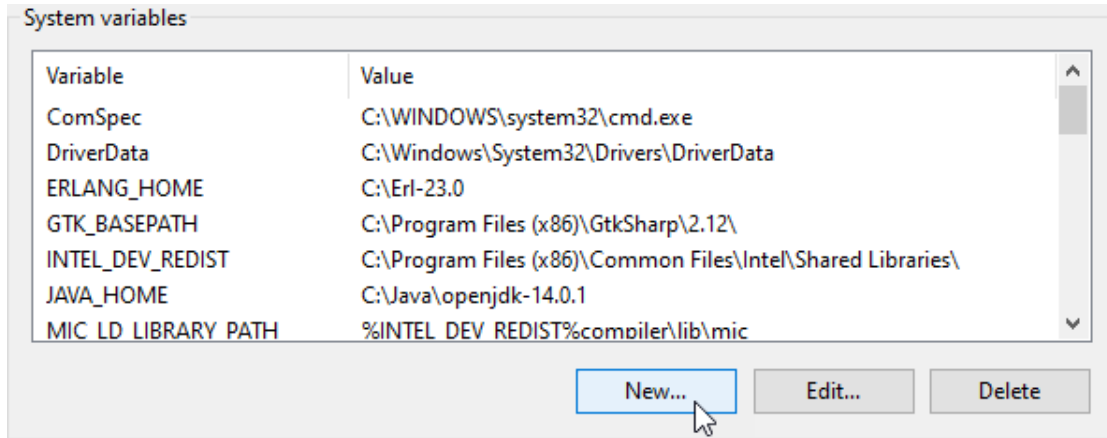
4. Click **Download R <version> for Windows**.
5. Run R-4.0.1-win.exe and follow the installation instructions.



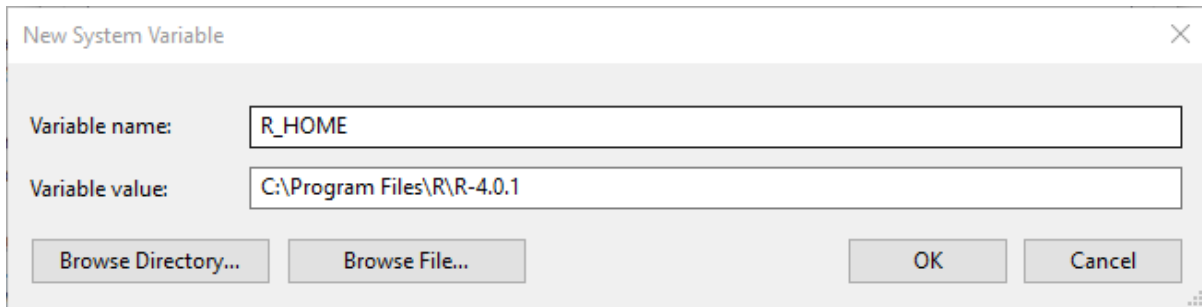
## Setting R Environment Variables

To set the R environment variables:

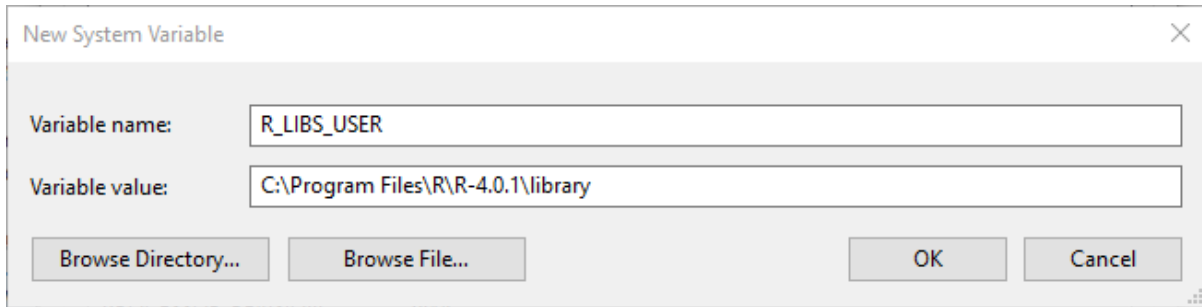
1. Go to **Control Panel** → **System** → **Advanced System Settings**.
2. Click the **Environment Variables** button.
3. Under **System Variables**, click: **New**.



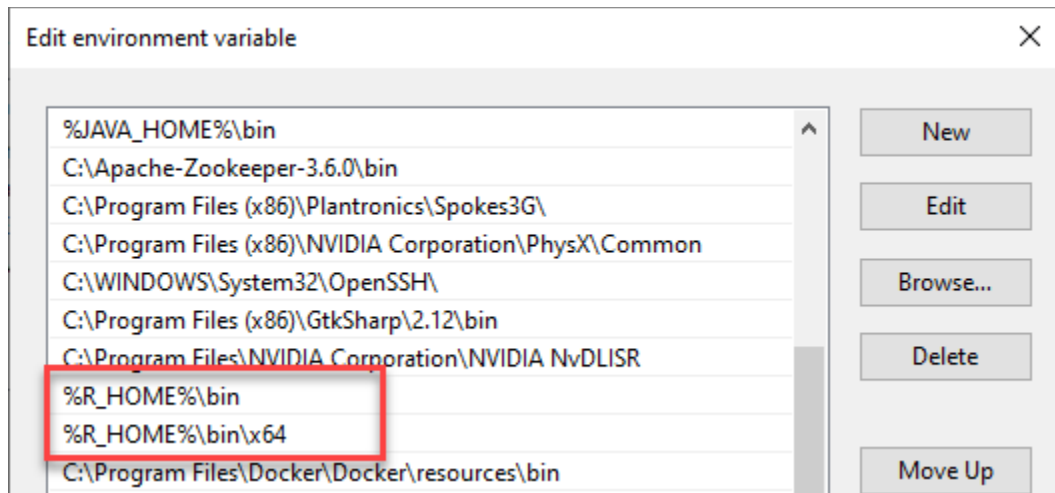
4. In the **Variable Name** field, enter `R_HOME`
5. Browse for the directory `C:\Program Files\R\R-4.0.1`



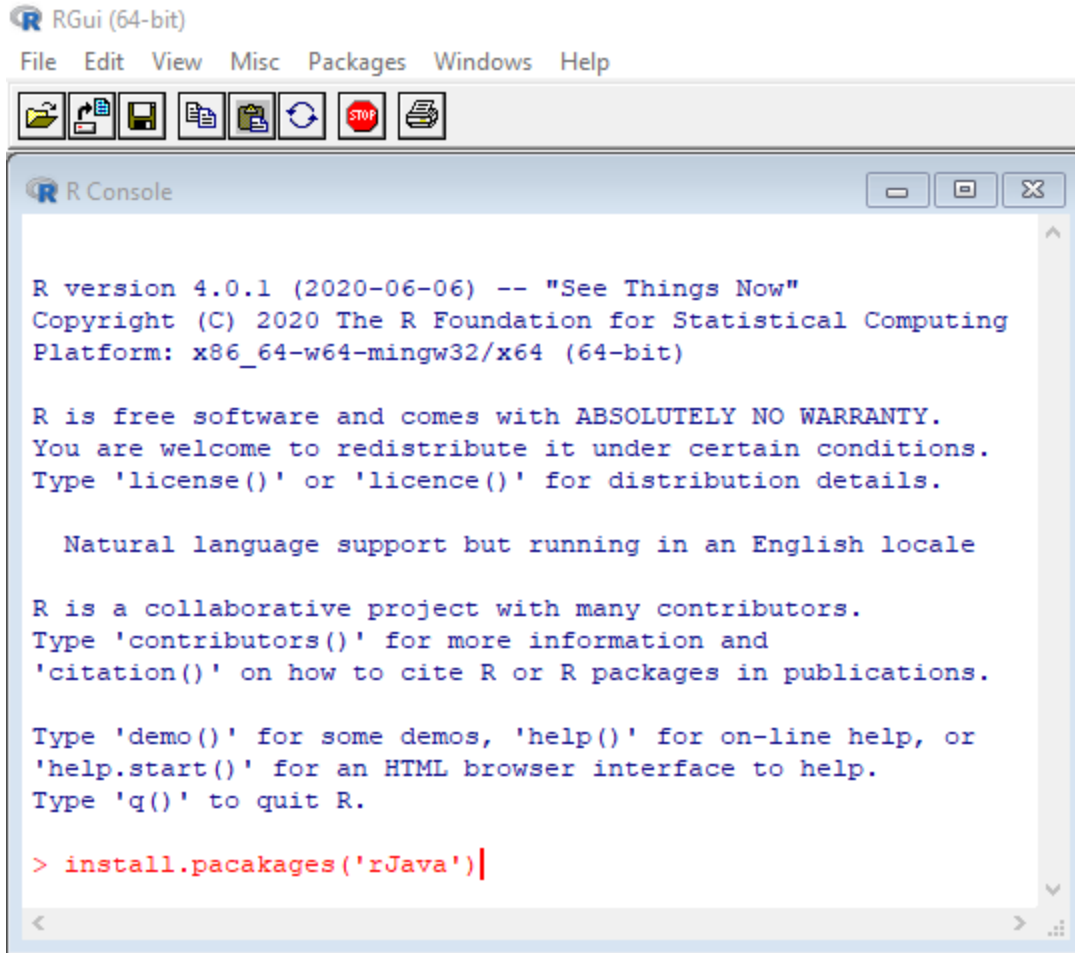
6. Repeat to add the variable `R_LIBS_USER`
7. Browse for the directory `C:\Program Files\R\R-4.0.1\library`



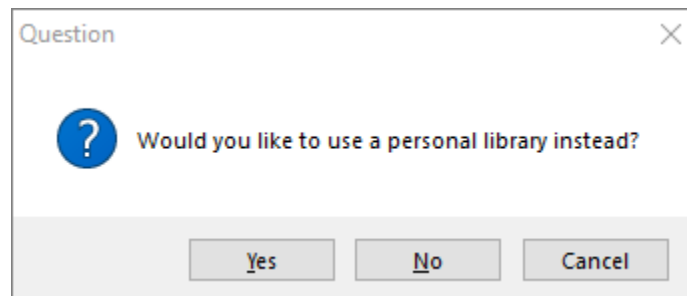
8. Add to the Path the location of the R executable: `%R_HOME%\bin\x64`
9. Ensure the path references `rcmd.exe` and `r.dll`.



10. Start R.
11. In the R Console, run the command `install.packages('rJava')`



12. If prompted with **Would you like to use a personal library instead?** click **Yes**.



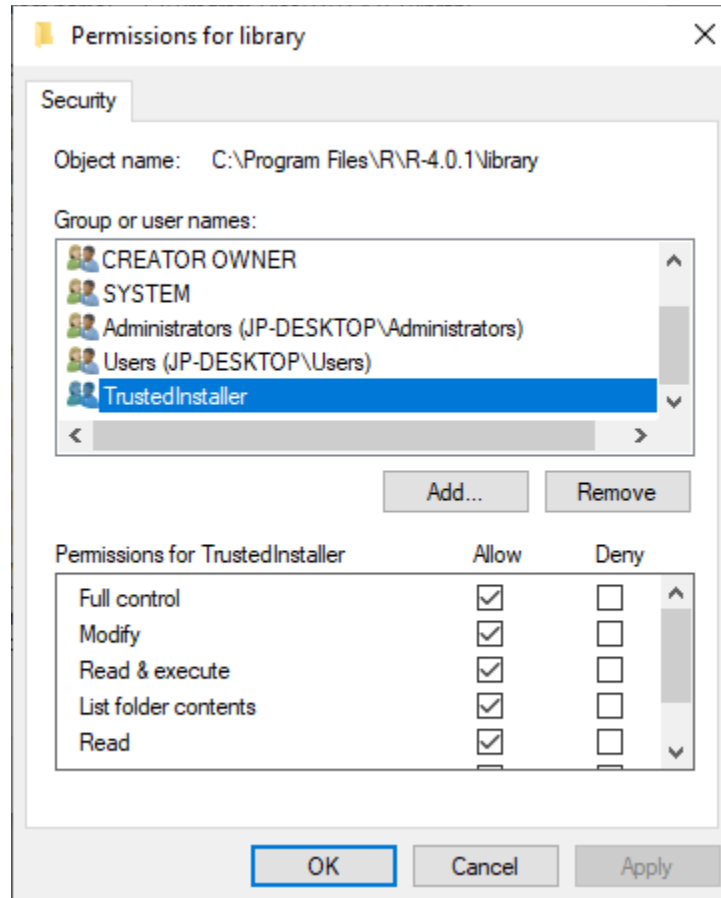
13. If prompted with the path of the library, click **Yes**.

14. When prompted for the CRAN mirror, choose a country, and then click **OK**.



*You may be denied permission writing to the library folder. You will need to change the permission for the folder.*

```
> install.packages('rJava')
Warning in install.packages("rJava") :
  'lib = "C:/Program Files/R/R-4.0.1/library"' is not writable
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.ma.imperial.ac.uk/bin/windows/contrib/4.0/rJava_0.9-12$
Content type 'application/zip' length 1501700 bytes (1.4 MB)
downloaded 1.4 MB
```



```
> install.packages('rJava')
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.ma.imperial.ac.uk/bin/windows/contrib/4.0/rJava_0.9-12$
Content type 'application/zip' length 1501700 bytes (1.4 MB)
downloaded 1.4 MB
```

```
package 'rJava' successfully unpacked and MD5 sums checked
```

```
The downloaded binary packages are in
```

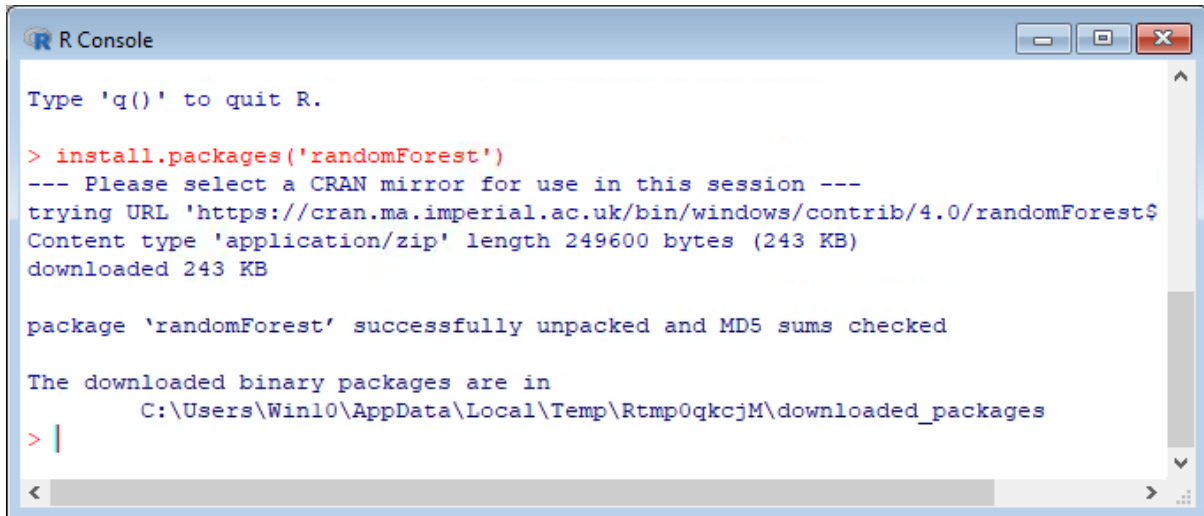
```
C:\Users\Jp\AppData\Local\Temp\RtmpwHwbG5\downloaded_packages
```

```
> |
```

## Installing the randomForest Library

To install the `randomForest` library:

1. Click on the R Console icon.
2. Enter the following command: `install.packages('randomForest')`



```
R Console

Type 'q()' to quit R.

> install.packages('randomForest')
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.ma.imperial.ac.uk/bin/windows/contrib/4.0/randomForest$
Content type 'application/zip' length 249600 bytes (243 KB)
downloaded 243 KB

package 'randomForest' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\Win10\AppData\Local\Temp\Rtmp0qkcjM\downloaded_packages
> |
```

3. After `randomForest` has successfully been installed, type `q()` to quit the R console.
4. Click **Yes** to close the workplace image.
5. Close R.

## Installing RStudio (Optional)

To install RStudio:

1. Download RStudio from [RStudio IDE](#).
2. Click on the **Download RStudio for Windows** button.

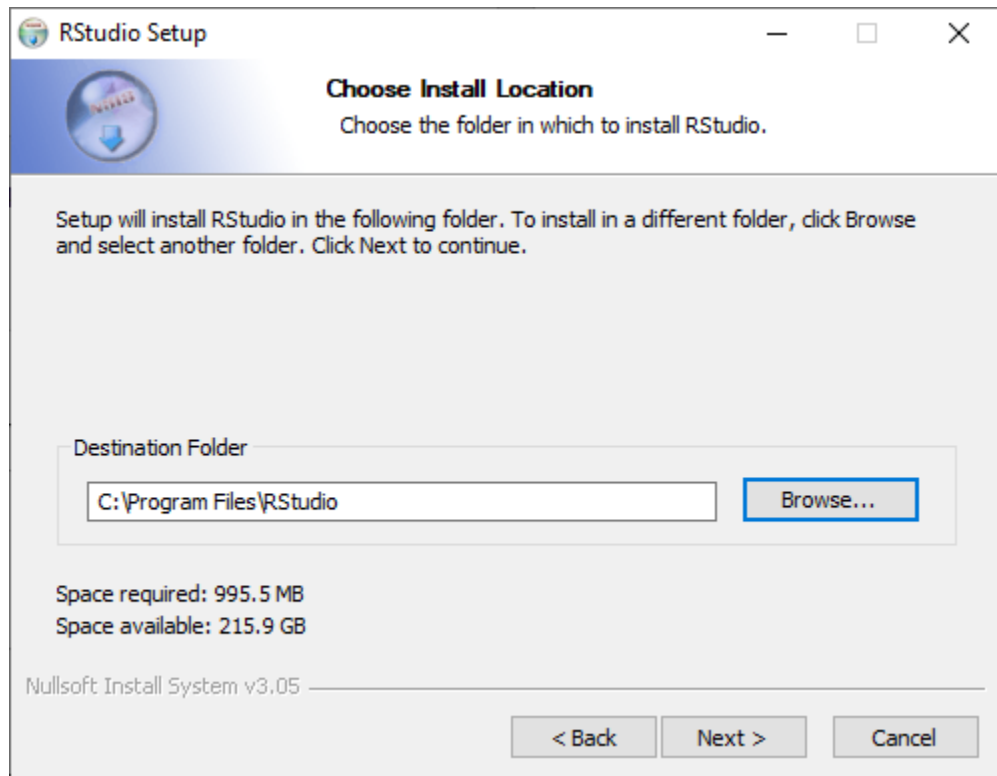
### RStudio Desktop 1.3.959 - [Release Notes](#)

1. Install R. RStudio requires [R 3.0.1+](#).
2. Download RStudio Desktop. Recommended for your system:



Requires Windows 10/8/7 (64-bit)

3. Run RStudio-1.3.959.exe and follow the installation instructions.





## Configuring Pentaho Data Integration with R

In the `rJava` directory, there is a `jri.dll` file that needs to be copied into the `libswt` directory of Spoon:

1. Stop Spoon, if it is running.
2. Find `%R_LIBS_USER%/rJava/jri/x64/jri.dll`
3. Copy `jri.dll` to the following directory:
  - a. Windows: `[Pentaho directory]/client-tools/data-integration/libswt/win64`
  - b. Linux: `[Pentaho directory]/client-tools/data-integration/libswt/linux`

Further details can be found in *R on PDI* in the [Pentaho Data Integration Best Practices library](#).

## Verifying Your Installation

1. Open a new transformation in PDI.
2. Drag an **RScript Executor** step onto the canvas.
3. Double-click the step and select the middle tab, **RScript**. You will see some comments at the top of the window:

---

```
# The main output is expected to be a data frame, unless "Output
# from script is text" is checked. So, to output a data frame the
# last statement in the script should be the name of the frame.
# In the case that the output is text (as would be seen on the
# R console), the last statement should be a "print" statement in
# order to print the object required.
```

---

4. Beneath the comment above, enter this code:

---

```
library(datasets)
iris
```

---

- Once you have entered this code in the **R Script** tab, click the **Test Script** button on this tab.

Examine preview data

Rows of step: R script executor (150 rows)

#	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Close Show Log

## Lab 1: Credit Card Fraud – AutoML

*Imagine that a direct retailer wants to reduce losses due to orders involving fraudulent use of credit cards. They accept orders via phone and their web site, and ship goods directly to the customer. Basic customer details, such as customer name, date of birth, billing address and preferred shipping address, are stored in a relational database.*

*Orders, as they come in, are stored in a database. There is also a report of historical instances of fraud contained in a CSV spreadsheet.*

### Objectives

In this guided demonstration, you will:

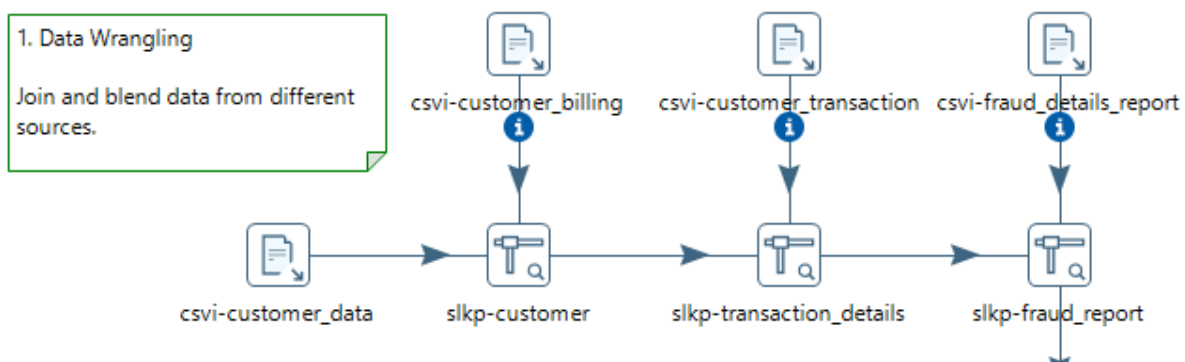
- Prepare Data
- Configure Python Executor step.
- Build and Train a Forest Tree Model.
- Deploy and Test the model.

### Step 1 – Data Preparation

With the goal of preparing a dataset for ML, we can use PDI to combine these disparate data sources and engineer some features for learning from it. The following figure shows a transformation demonstrating an example of just that, and includes some steps for deriving new fields.

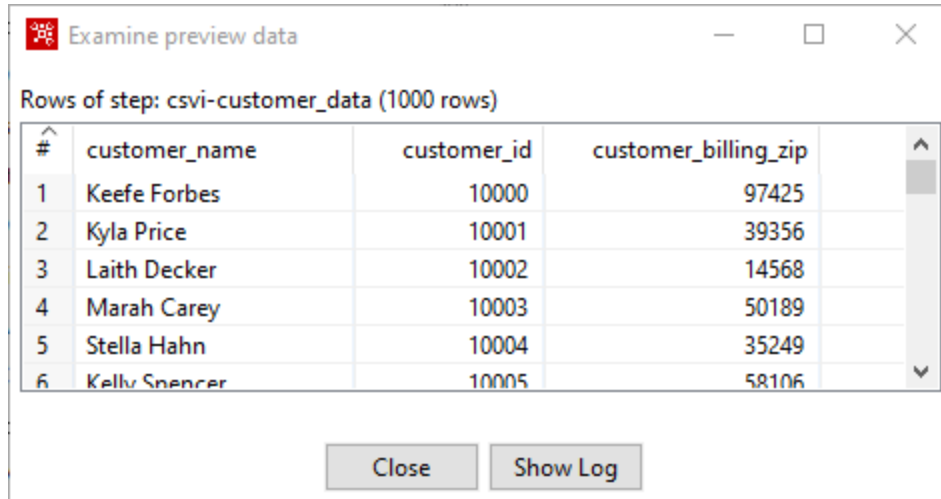
To begin with, customer data is joined from several data sources, and then blended with transactional data and historical fraud occurrences contained in a CSV file.

1. In Spoon, open the following main job: C:\Machine--  
Learning\01\_Credit\_Card\Lab\_01\_AutoML\tr\_autoML.ktr



2. Browse the various customer data sources:

`csvi-customer_data`, where you will find the `customer_billing_zip` codes, which will be used in feature engineering:

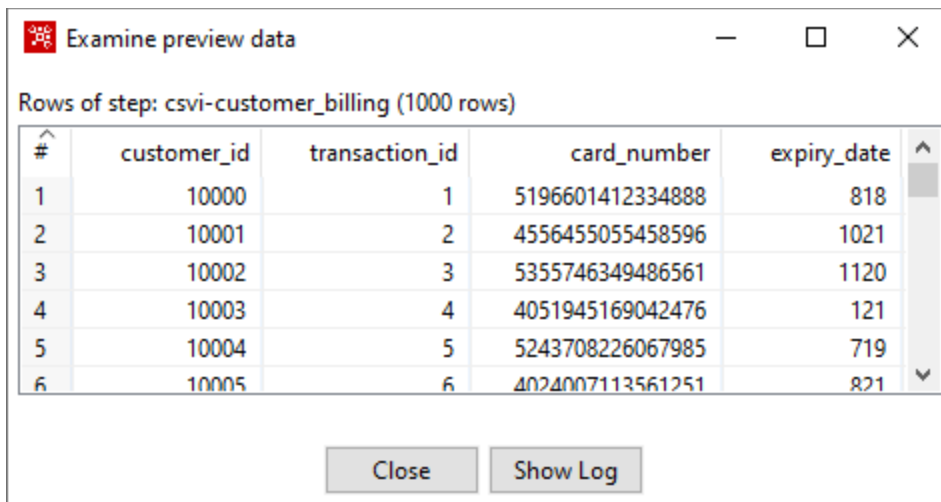


Rows of step: csvi-customer\_data (1000 rows)

#	customer_name	customer_id	customer_billing_zip
1	Keefe Forbes	10000	97425
2	Kyla Price	10001	39356
3	Laith Decker	10002	14568
4	Marah Carey	10003	50189
5	Stella Hahn	10004	35249
6	Kelly Spencer	10005	58106

Close Show Log

`csvi-customer_billing`, which references the customer transaction:



Rows of step: csvi-customer\_billing (1000 rows)

#	customer_id	transaction_id	card_number	expiry_date
1	10000	1	5196601412334888	818
2	10001	2	4556455055458596	1021
3	10002	3	5355746349486561	1120
4	10003	4	4051945169042476	121
5	10004	5	5243708226067985	719
6	10005	6	4024007113561251	821

Close Show Log

`csvi-customer_transaction`:

- Customer transaction details
- Feature engineering for `ship_to_zip`
- The transaction details (x variables) are used by the decision trees to determine whether the transaction is fraudulent (y variable). The Boolean values will need to be changed into numbers for the `randomForest` algorithm.

Examine preview data

Rows of step: csvi-customer\_transaction (1000 rows)

ship_to_zip	first_time_customer	order_dollar_amount	num_items	age	web_order
97790	N	964.7	19	104.7	Y
39356	N	524.1	15	24.2	N
14773	N	234.2	18	13.8	Y
50189	Y	207.7	8	56.3	N
35249	Y	910.6	11	59.1	N
58209	Y	837.3	4	73.7	N
47091	Y	724.4	11	108.7	N

Close Show Log

csvi-fraud\_details\_report, which indicates whether historically the transaction was fraudulent:

Examine preview data

Rows of step: csvi-fraud\_details\_report (1000 rows)

#	transaction_id	reported_as_fraud_historic
1	1	N
2	2	N
3	3	N
4	4	N
5	5	Y
6	6	Y

Close Show Log

## Step 2 – Feature Engineering

The Feature Engineering is set to `billing zip code = shipping zip code`.

1. Open the step **frmla-billing=shipping**.

The screenshot shows the 'fx Formula' dialog box. The 'Step name' field is set to 'frmla-billing=shipping'. Below it, a table titled 'Fields:' contains one entry:

#	New field	Formula	Value type
1	billing_shipping_zip_equal	[customer_billing_zip] = [ship_to_zip]	Boolean

At the bottom left, the 'Help' button is highlighted with a blue box. Other buttons include 'OK' and 'Cancel'.

There are steps for deriving additional fields that might be useful for predictive modeling. These include computing the customer's age, extracting the hour of the day the order was placed, and setting a flag to indicate whether the shipping and billing addresses have the same zip code.

## Step 3 – Test Machine Learning Models to Identify the Most Accurate Model

So, what does the data scientist do at this point?

Typically, they will want to get a feel for the data by examining simple summary statistics and visualizations, followed by applying quick techniques for assessing the relationship between individual attributes (fields) and the target of interest which, in this example, is the `reported_as_fraud_historic` field.

Following that, if there are attributes that look promising, quick tests with common supervised classification algorithms will be next on the list. This comprises the initial stages of experimental data mining – that is, the process of determining which predictive techniques are going to give the best result for a given problem.

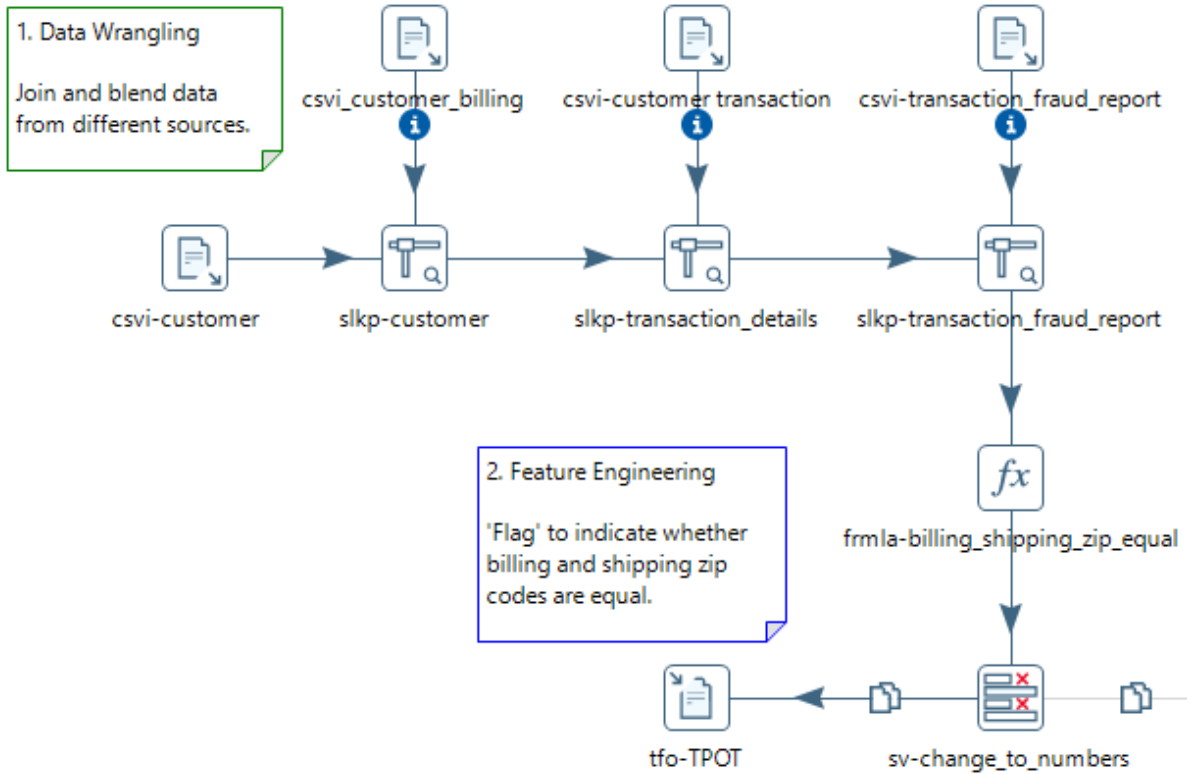
### TPOT: *py-auto\_ml*

A Tree-based Pipeline Optimization Tool for Automating Machine Learning (TPOT) is a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming. TPOT will automate the most tedious part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for your data.

One of the most useful tools for writing and executing Python script is Google Colab.

1. Run the transformation C:\Machine--

Learning\01\_Credit\_Card\Lab\_01\_AutoML\tr\_autoML.ktr



2. Open the file C:\Machine--

Learning\01\_Credit\_Card\Lab\_01\_AutoML\data\TPOT.csv

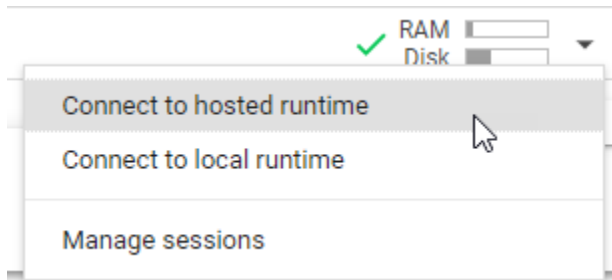
TPOT.csv	
1	0;964.7;19;104.7;1;40;7;0;0
2	0;524.1;15;24.2;0;55;4;1;0
3	0;234.2;18;13.8;1;66;15;0;0
4	1;207.7;8;56.3;0;45;23;1;0
5	1;910.6;11;59.1;0;31;23;1;1

This will be the dataset used for autoML in Colab.

## Colab

The following code overviews the steps:

1. In a Chrome browser, open  
`https://colab.research.google.com/notebooks/intro.ipynb`
2. Ensure you have connected to the hosted runtime.



If you wish to browse the finished script:

1. Click on **File → Upload notebook**.  
  
C:\Machine--  
Learning\01\_Credit\_Card\Lab\_01\_AutoML\scripts\credit\_card\_fraud.ipynb
2. Save the Notebook using **File → Save**



*We recommend that you save either to GitHub or Google Drive.*



## AutoML Script

These are the code sections for the Jupyter file `credit card fraud.ipynb`:

1. Install the TPOT libraries:

---

```
# Installs TPOT libraries.
!pip install tpot
```

---

2. Run the step:

### ▼ Install TPOT

```
1 # Installs TPOT libraries.
2 !pip install tpot
```

Collecting tpot  
 Downloading <https://files.pythonhosted.org/packages/14/5e/cb87b0257033a7a396e53>  
 [██] 92kB 3.5MB/s  
 Collecting stopit>=1.1.1  
 Downloading <https://files.pythonhosted.org/packages/35/58/e8bb0b0fb05baf07bbac1>  
 Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.6/dist-pa  
 Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.6/d  
 Collecting update-checker>=0.16  
 Downloading <https://files.pythonhosted.org/packages/d6/c3/aaf8a162df8e8f9d32123>  
 Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.6/dist-pa  
 Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.6/dist-pack  
 Collecting deap>=1.2  
 Downloading <https://files.pythonhosted.org/packages/0a/eb/2bd0a32e3ce757fb26264>  
 [██] 163kB 7.4MB/s  
 Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.6/dist-pack  
 Requirement already satisfied: numpy>=1.16.3 in /usr/local/lib/python3.6/dist-pac  
 Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.6/dist-p  
 Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6  
 Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-pack  
 Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/lo  
 Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dis  
 Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-pack  
 Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist  
 Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages  
 Building wheels for collected packages: stopit  
 Building wheel for stopit (setup.py) ... done  
 Created wheel for stopit: filename=stopit-1.1.2-cp36-none-any.whl size=11956 sh  
 Stored in directory: /root/.cache/pip/wheels/3c/85/2b/2580190404636bfc63e8de3df  
 Successfully built stopit  
 Installing collected packages: stopit, update-checker, deap, tpot  
 Successfully installed deap-1.3.1 stopit-1.1.2 tpot-0.11.5 update-checker-0.17

### 3. Import libraries:

---

```
import numpy as np
import pandas as pd
from tpot import TPOTClassifier
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
```

---

### 4. Import dataset from C:\Machine--

Learning\01\_Credit\_Card\Lab\_01\_AutoML\data\TPOT.csv:

---

```
from google.colab import files
uploaded = files.upload()
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))

dataset = pd.read_csv('TPOT.csv', sep= ';', header=None)
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 8].values
```

---

### 5. Add column headers:

---

```
dataset.columns =
['first_time_customer', 'order_dollar_amount', 'num_items', 'age', 'web_order',
'total_transactions_to_date', 'hour_of_day', 'billing_shipping_zip_equal', 're
ported_as_fraud_historic']
```

---

### 6. Convert dataset to numpy array and fit data (optional):

---

```
x = dataset.iloc[:, 0:-1].values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
X=np.asarray(x_scaled)
y=np.asarray(dataset.iloc[:, -1])
```

---

### 7. Split the dataset. 75% used for test.

---

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.75,
random_state=None)
```

---

## 8. Run TPOT Classifier:

---

```
tpot = TPOTClassifier(generations=1, verbosity=2, population_size=100,
scoring='accuracy', n_jobs = -1, config_dict='TPOT light')

tpot.fit(X_train, y_train)

output_score=str(tpot.score(X_test, y_test))

print(tpot.fitted_pipeline_)
```

---

## 9. Export Pipeline as Python script:

---

```
tpot.export('tpot_exported_credit_card_pipeline.py')

from google.colab import files

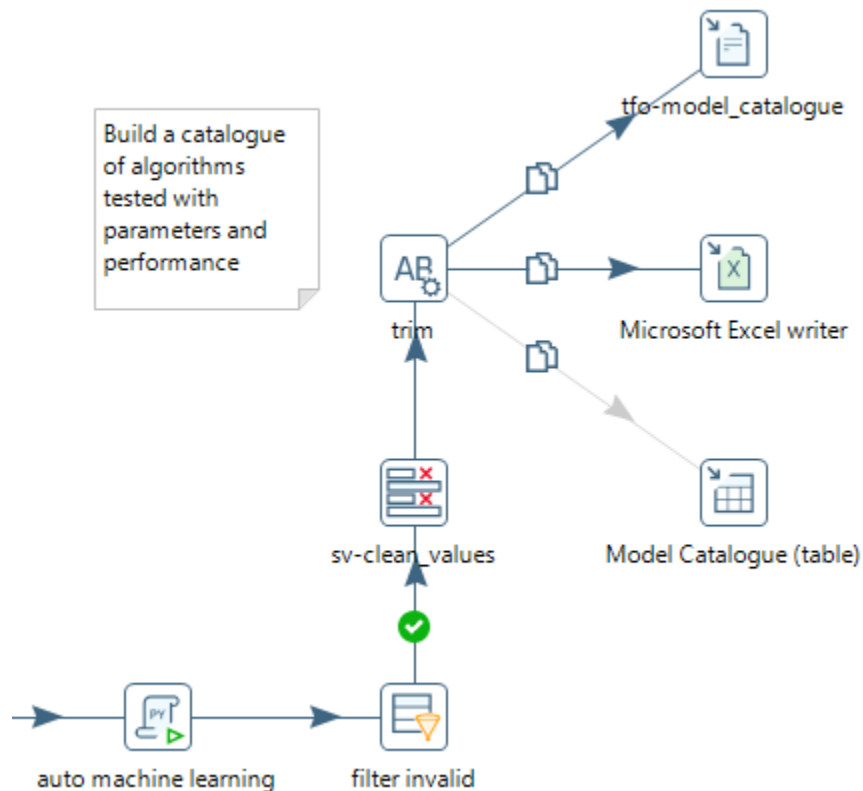
files.download('tpot_exported_credit_card_pipeline.py')
```

---

## Python Executor

The script has been tested and is now ready for deployment in PDI.


1. Enable the rest of the hops in the transformation, except: **Model Catalogue (table)**:



2. Open the step **py-auto\_ml**

3. Ensure you set the path to Python:

☒ Use a Python virtual environment

C:\Program Files\Python\Python38\python.exe  Browse...



For further details on the script, see [Appendix A](#).

To ensure the script does not take a long time to process, the following TPOT parameters have been set:

```
tpot = TPOTClassifier(generations=1, verbosity=2, population_size=100,
config_dict='TPOT light')
```



For further details on TPOT parameters, see [Appendix B](#).

4. Click on the **Input** tab.
  - a. Use this tab to make selections for moving data from PDI fields to Python variables.
  - b. The **All rows** option is commonly used for data frames. A data frame is used for storing data tables and is composed of a list of vectors of equal length.
  - c. Select the **All rows** option to process all your data at once, for example, using the Python list of dictionaries.

Option	Description
<b>Available variables</b>	Use the plus sign button to add a Python variable to the input mapping for the script used in the transformation. You can remove the Python variable by clicking the X icon.
<b>Variable name</b>	Enter the name of the Python variable. The list of available variables will update automatically.
<b>Step</b>	Specify the name of the input step to map from. It can be any step in the parent transformation with an outgoing hop connected to the <b>Python Executor</b> step.

Option	Description
<b>Data structure</b>	<p>Specify the data structure from which you want to pull the fields for mapping. You can select one of the following:</p> <ul style="list-style-type: none"> <li>• <b>Pandas data frame:</b> the tabular data structure for Python/Pandas.</li> <li>• <b>NumPy array:</b> the table of values, all the same type, which is indexed by a tuple of positive integers.</li> <li>• <b>Python List of Dictionaries:</b> each row in the PDI stream becomes a Python dictionary. All the dictionaries are put into a Python list.</li> </ul>

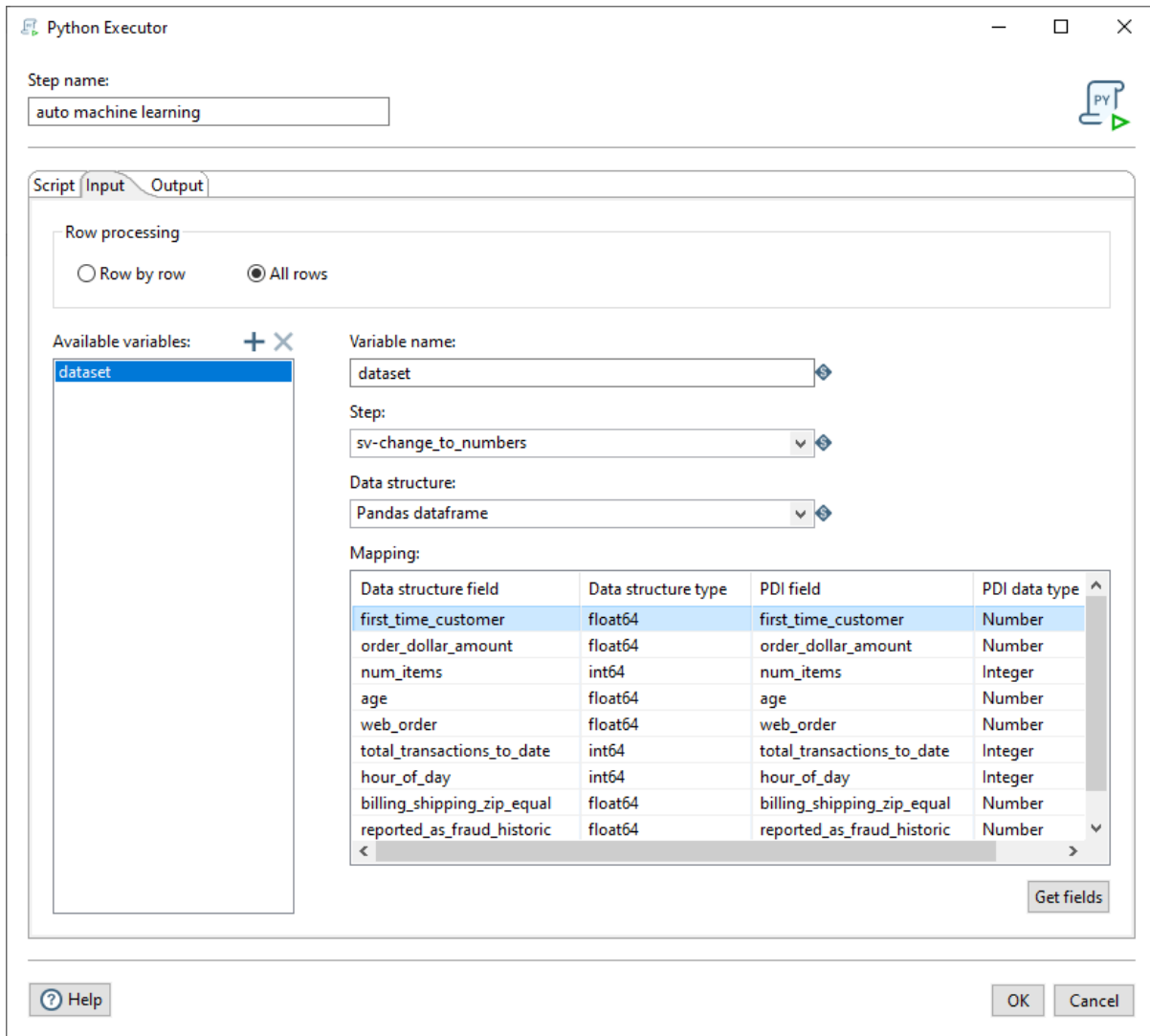
5. The Mapping table contains the following field properties:

Field Property	Description
<b>Data structure field</b>	The value of the Python data structure field to which you want to map the PDI field.
<b>Data structure type</b>	The value of the data structure type assigned to the data structure field to which you want to map the PDI field.
<b>PDI field</b>	The name of the PDI field which contains the vector data stored in the mapped Python variable.
<b>PDI data type</b>	The value of the data type assigned to the PDI field, such as a date, a number, or a timestamp.

6. Select the **Get fields** button to populate the table with fields from the input step(s) in your transformation. If necessary, you can modify your selections.



For further details, see [Python Executor](#).



- The `cust` variable defines the dataframe in the Python script using `iloc`:

---

```
x = dataset.iloc[:, 1:-1].values
```

---

- The dataframe is pulled from the PDI step **sv-changes\_to\_numbers**.

From this list, for the purposes of predictive modeling, we can drop the customer name, ID fields, email addresses, phone numbers and physical addresses. These fields are unlikely to be useful for learning purposes and, in fact, can be detrimental due to the large number of distinct values they contain.

7. Click on the **Output** tab.
8. The output of `model_df` dataframe, from the script:

---

```
model_df=pd.DataFrame(model_list,columns=['pipe','generation','mutation','c  
rossover','predecessor','operator','cv'])
```

---

is converted back to PDI fields.

9. Examine the Logging.

10. Sort by **Cross\_Validation\_Performance**:

Iteration	Cross_Validation_Performance	ML_Algorithms_and_Parameters
94	0.8447470942	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=entropy, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)
30	0.8439470942	DecisionTreeClassifier(MinMaxScaler(input_matrix), DecisionTreeClassifier__criterion=entropy, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)
96	0.841149499	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=gini, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)
4	0.840349499	DecisionTreeClassifier(MaxAbsScaler(input_matrix), DecisionTreeClassifier__criterion=gini, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)
155	0.839949499	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=gini, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)
156	0.8399462926	DecisionTreeClassifier(MaxAbsScaler(input_matrix), DecisionTreeClassifier__criterion=gini, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)
120	0.8391470942	DecisionTreeClassifier(MinMaxScaler(input_matrix), DecisionTreeClassifier__criterion=gini, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)
142	0.8391462926	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=gini, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)
41	0.8387382766	KNeighborsClassifier(StandardScaler(input_matrix), KNeighborsClassifier__n_neighbors=5, KNeighborsClassifier__weights='uniform')
0	0.8379511022	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=gini, DecisionTreeClassifier__max_depth=8, DecisionTreeClassifier__min_samples_leaf=11, DecisionTreeClassifier__min_samples_split=8)

Generation 1 - Current best internal CV score: 0.8447470941883767

Best pipeline: DecisionTreeClassifier(input\_matrix, criterion=entropy, max\_depth=8, min\_samples\_leaf=11, min\_samples\_split=8)  
0.8474666666666667

The output from the TPOT is tidied up before writing the results to a model catalogue file.

- Filter – removes invalid results
- Select values – orders and renames some of the output fields
- String Operations – trims data stream field
- Text file output – output results to the text file `model_catalogue.txt`
- Microsoft Excel Writer – output results to the Excel workbook `model_catalogue.xlsx`

### What does this mean?

For the First Generation, the best algorithm pipeline run is DecisionTree with a scoring of 0.844 and accuracy of 0.84746 (figure used to judge the quality of the pipeline).

11. Open the Excel file C:\Machine--

Learning\01\_Credit\_Card\Lab\_01\_AutoML\output\model\_catalogue.xlsx

Iteration	Cross_Validation_Performance	ML_Algorithms_and_Parameters
94	0.844747094	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=entropy,
30	0.843947094	DecisionTreeClassifier(MinMaxScaler(input_matrix), DecisionTreeClassifier__c
96	0.841149499	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=gini, D
4	0.840349499	DecisionTreeClassifier(MaxAbsScaler(input_matrix), DecisionTreeClassifier__c
155	0.839949499	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=gini, D
156	0.839946293	DecisionTreeClassifier(MaxAbsScaler(input_matrix), DecisionTreeClassifier__c
120	0.839147094	DecisionTreeClassifier(MinMaxScaler(input_matrix), DecisionTreeClassifier__c
142	0.839146293	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=gini, D
41	0.838738277	KNeighborsClassifier(StandardScaler(input_matrix), KNeighborsClassifier__n_
0	0.837951102	DecisionTreeClassifier(input_matrix, DecisionTreeClassifier__criterion=gini, D
77	0.837142285	KNeighborsClassifier(input_matrix, KNeighborsClassifier__n_neighbors=6, KN

### Conclusion

The best pipeline to use (with 85% accuracy) for this dataset is based on Decision Trees with a minimum of eight trees.

It may also be worth looking at KNeighbors Classifier.

The object of using TPOT is to point you in the right direction for selecting the appropriate algorithm.



*The results will be different each time you run the TPOTClassifier.*



## Lab 2: Credit Card Fraud – randomForest

*The results from TPOT point to using a Decision Tree algorithm.*

### Objectives

In this guided demonstration, you will:

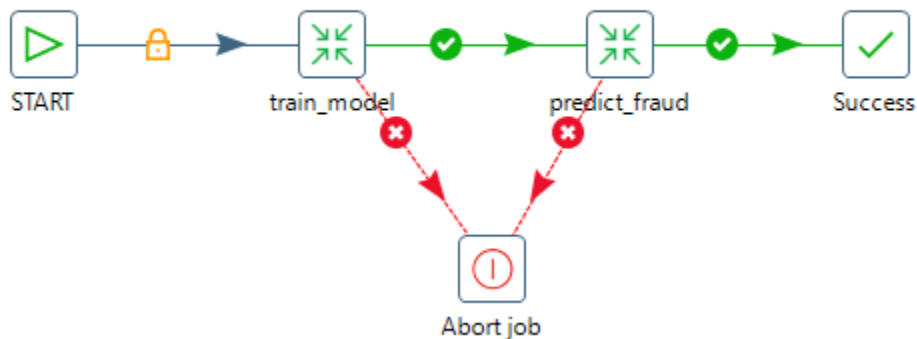
- Train a `randomForest` model in R.
- Deploy your model.
- Predict fraudulent credit card transactions.

The model that will be used is `randomForest`.

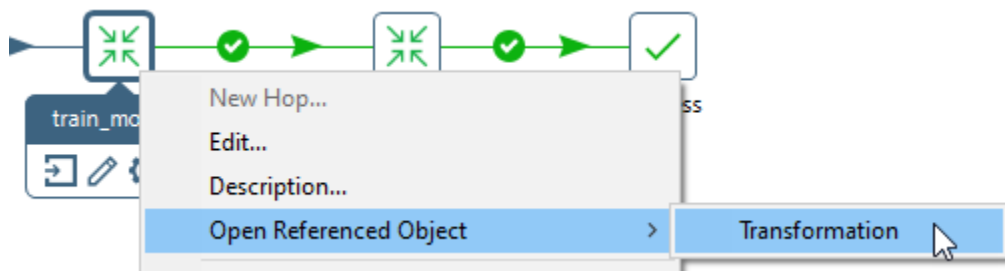
### Train the Model

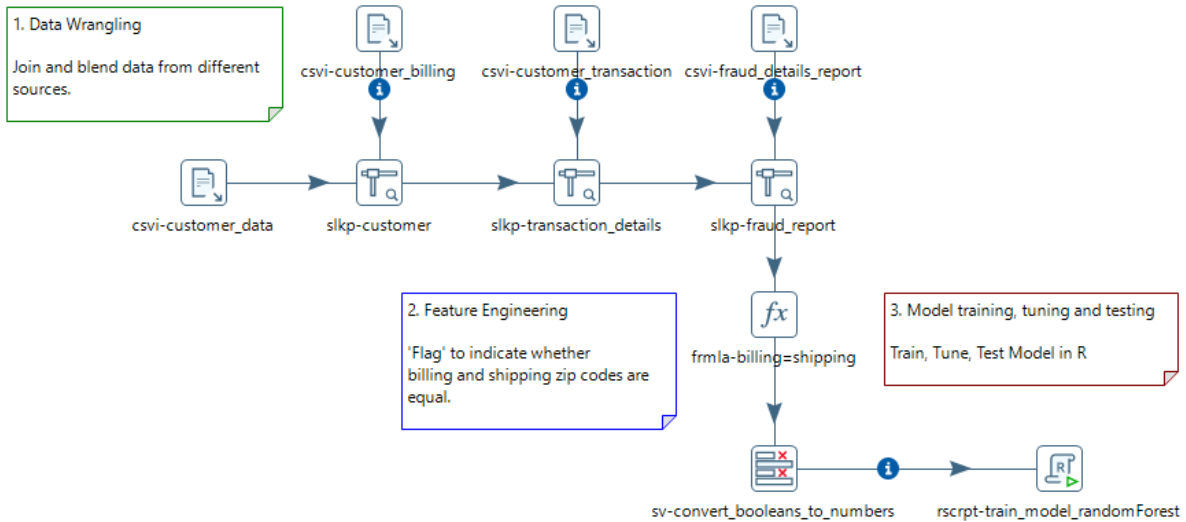
1. In Spoon, open the following main job: `C:\Machine--  
Learning\01_Credit_Card\Lab_02_Credit_Card_Fraud\jb_fraud_main_job.kjb`

This job will train a Model using Random Forest and Predict new values.



2. Let's look at the transformation that trains for the model. Right-click on the **train\_model** transformation and select **Open Referenced Object** → **Transformation**.





For an overview of the steps, see the previous [Lab 1: Credit Card Fraud – AutoML](#).

## R Script Executor: Train Model

1. Double-click on the **rsrpt-train\_model\_randomForest** step to bring up the configuration settings.
2. Under the **Configure** tab, ensure the **Input Frames** points to the step name **sv-convert\_booleans\_to\_numbers** and the R Frame name **train**.

**Row Handling**

Number of Rows to Process: All   
 Reservoir Sampling: ☐  
 Random seed: 1   
 Size:   
 Size:

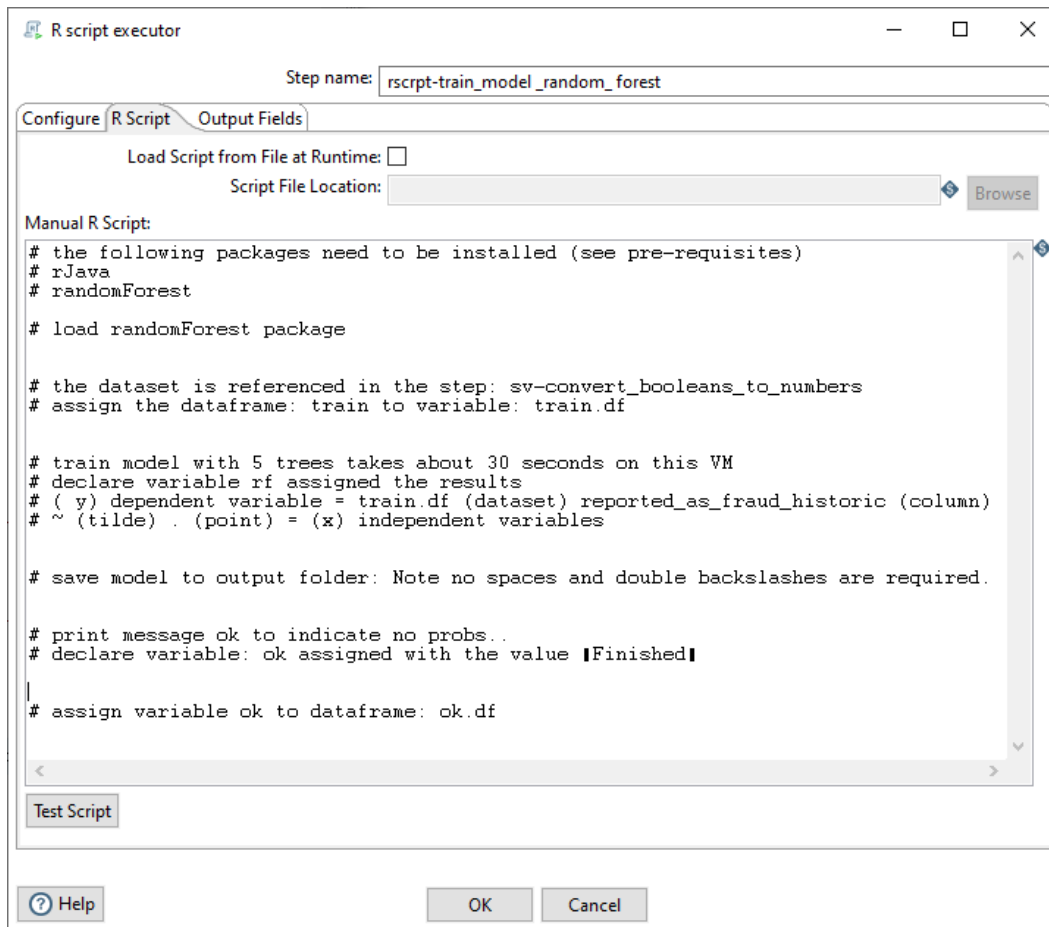
**Options**

Strings as Factors in R: ☒  
 Output from Script is Text: ☐  
 Include Input Fields as Output: ☐  
 Logging Level for R Messages: Basic

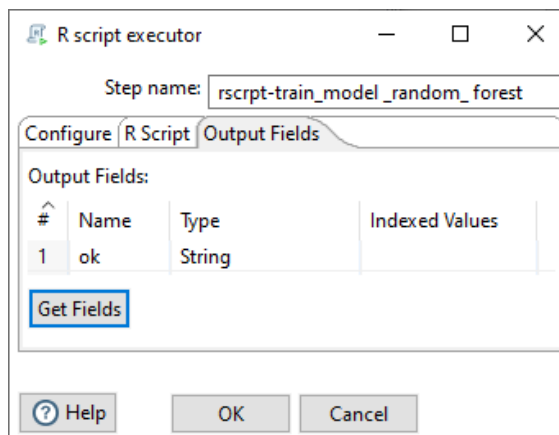
**Input Frames:**

#	Step name:	R Frame name
1	sv-convert_booleans_to_numbers	train

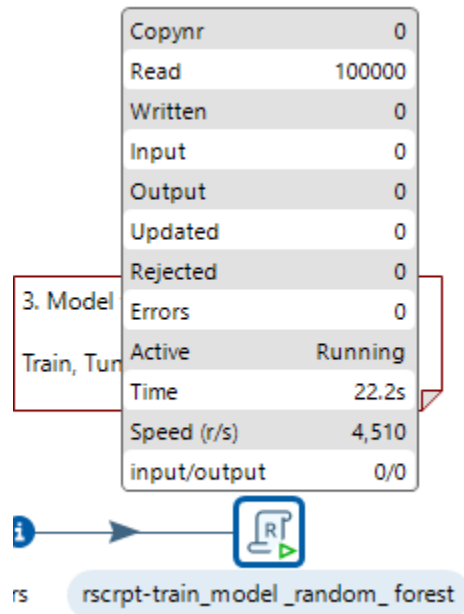
3. Set **Row Handling** to **Number of Rows to Process: All**.
4. Select the **R script** tab. Copy and paste the code snippets based on the Comments.



5. The required script is located at C:\Machine--  
Learning\Lab\_02\_Credit\_Card\_Fraud\scripts\train\_model.txt
6. Click on the **Output** tab.



- Run the transformation.

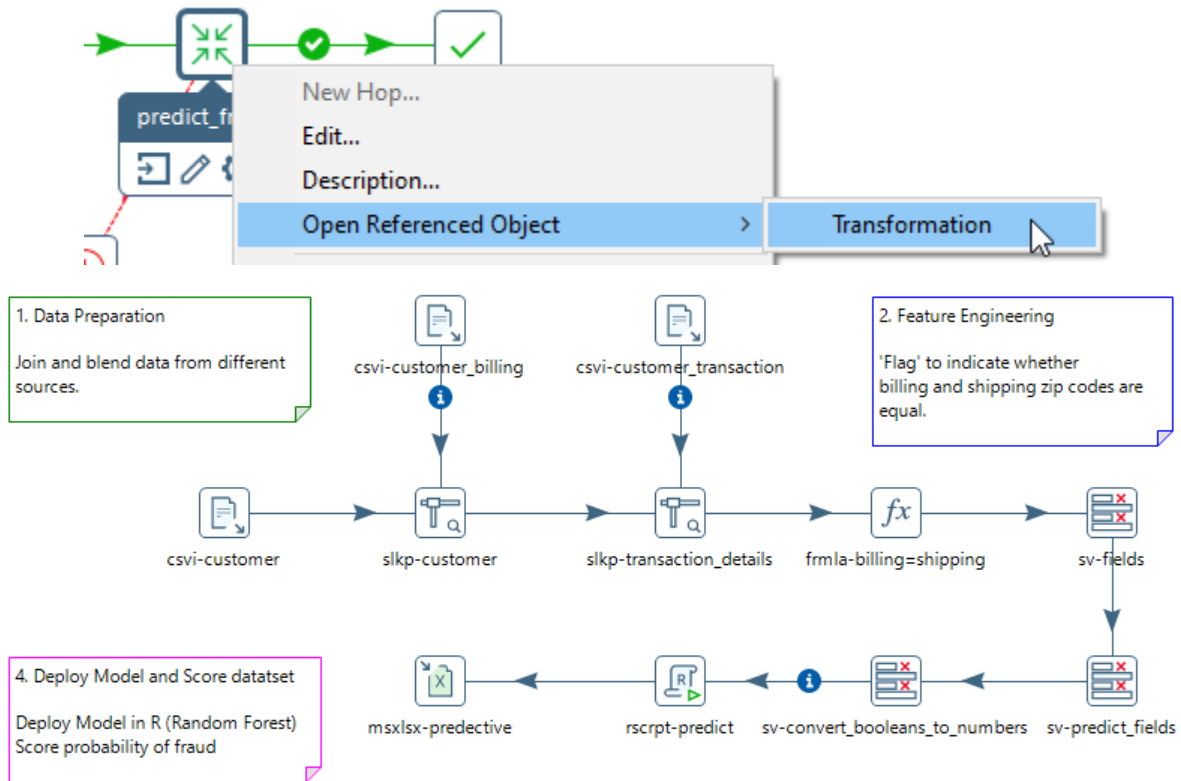


- Check that the model has been saved in C:\Machine--  
Learning\01\_Credit\_Card\Lab\_02\_Credit\_Card\_Fraud\scripts\rf.rdata

## Predict Fraudulent Credit Card Transactions

Let's look at the transformation that predicts fraudulent credit card activity based on our trained model.

1. Right-click on the **predict\_model** transformation and select **Open Referenced Object** → **Transformation**.



2. Double-click on the **rscript-predict** step to bring up the configuration settings.
3. Under the **Configure** tab, ensure the **Input Frames** points to the step name **sv-convert\_booleans\_to\_numbers** and that the **R Frame** name is **test**.

R script executor

Step name:

Configure R Script Output Fields

Row Handling

Number of Rows to Process:  Size:

Reservoir Sampling: ☒ Size:

Random seed:

Options

Strings as Factors in R: ☒

Output from Script is Text: ☐

Include Input Fields as Output Fields: ☐

Logging Level for R Messages:

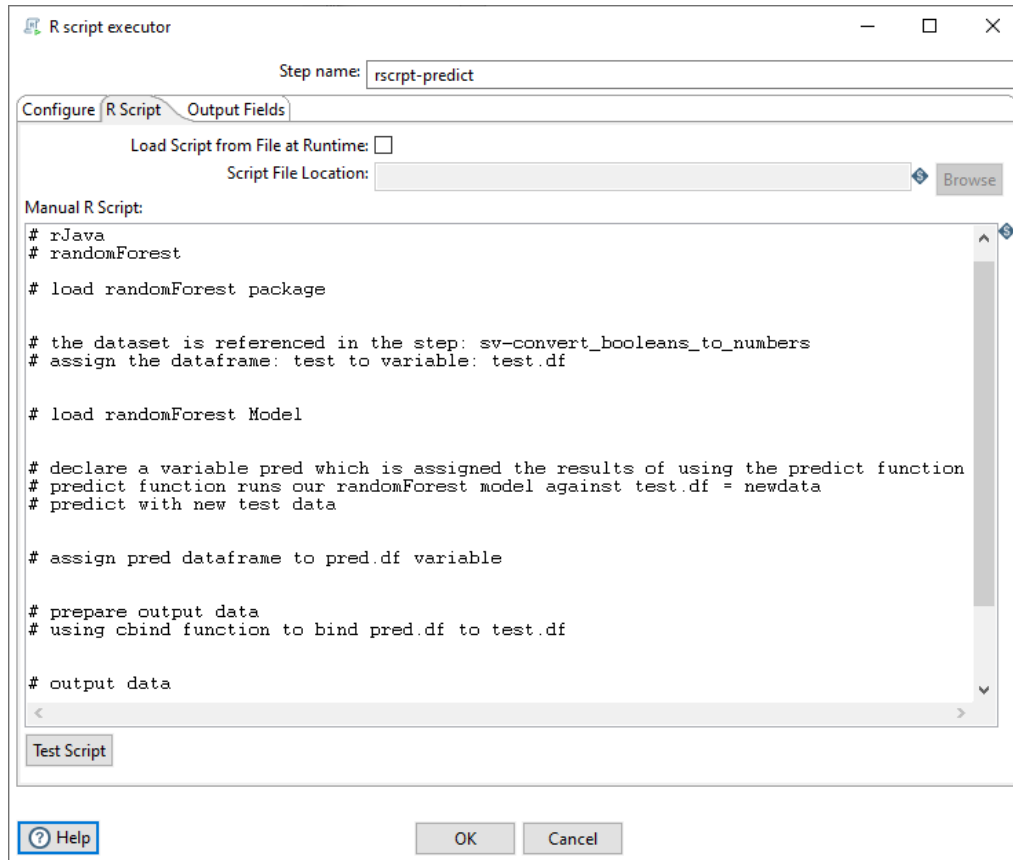
Input Frames:

#	Step name:	R Frame name
1	sv-convert_booleans_to_numbers	test

[? Help](#)

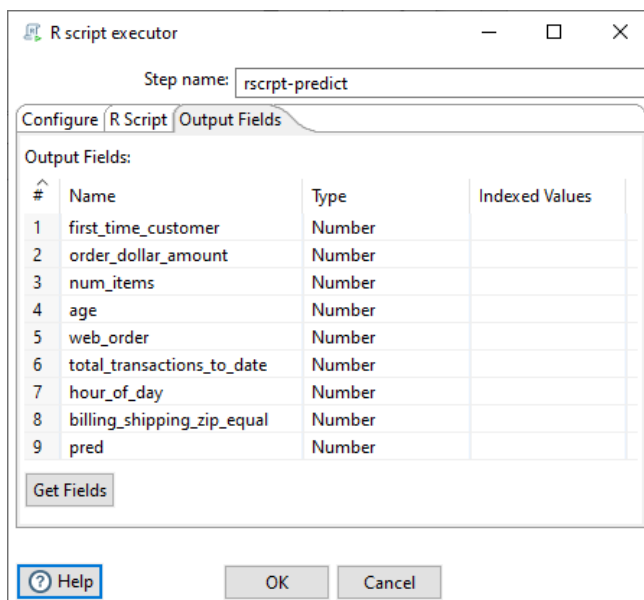
To remove any bias from the dataset, the complete dataset is randomly sampled (mixed up) .

4. Select the **Rscript** tab. Copy and paste the code snippets based on the Comments.

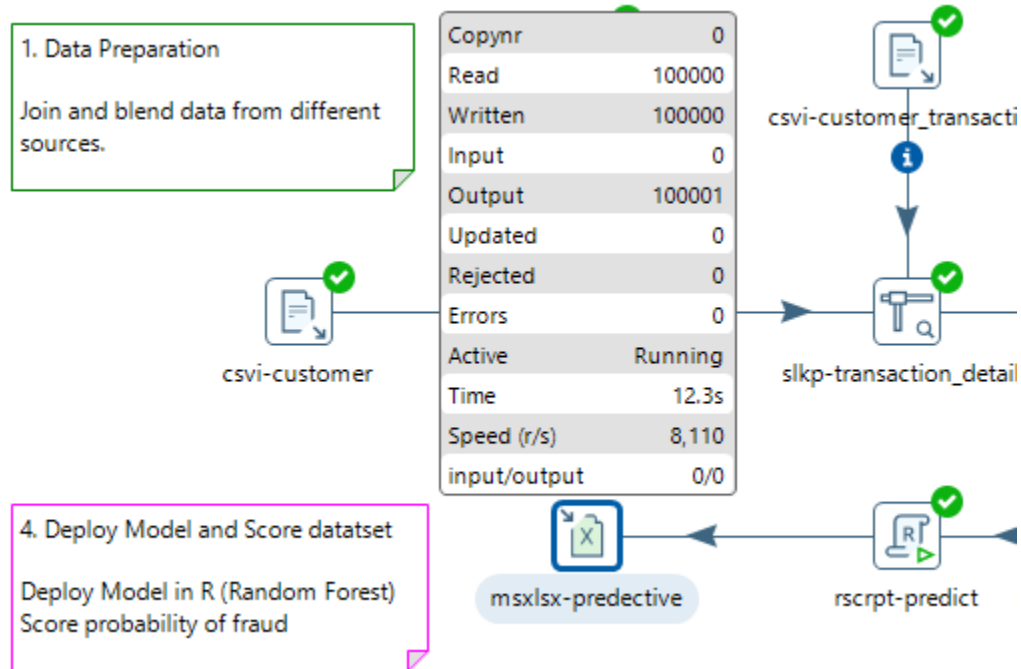


5. The required script is located at C:\Machine--  
Learning\01\_Credit\_Card\Lab\_02\_Credit\_Card\_Fraud\scripts\predict\_model.  
txt

6. Click on the **Output** tab.



7. Run the transformation.



8. Ensure all the steps have completed.

9. Open the Excel workbook C:\Machine--

Learning\01\_Credit\_Card\Lab\_02\_Credit\_Card\_Fraud\output\credit\_card\_predict.xlsx

transaction_id	card_number	first_time_customer	order_dollar_amount	num_items	age	web_order	total_transactions_to_date	hour_of_day	billing_shipping_zip_equal	pred
1	5196601412334890	0	964.7	19	104.7	1	40	7	0	12%
2	4556455055458600	0	524.1	15	24.2	0	55	4	1	1%
3	5355746349486560	0	234.2	18	13.8	1	66	15	0	30%
4	4051945169042480	1	207.7	8	56.3	0	45	23	1	1%
5	5243708226067980	1	910.6	11	59.1	0	31	23	1	92%
6	4024007113561250	1	837.3	4	73.7	0	64	21	0	97%
7	5472379586997450	1	724.4	11	108.7	0	72	2	1	0%
8	6011580060751470	1	354.8	12	39.2	1	75	9	0	91%
9	5447028309864970	0	730.8	18	14.4	0	87	1	1	20%
10	4552456819702940	1	338.7	13	41.5	0	46	11	0	12%
11	4716835573702620	0	677.1	15	84.3	1	59	2	1	14%
12	5524869937605800	1	307.5	18	73	1	20	3	1	100%



The complete solution can be found at C:\Machine--  
Learning\01\_Credit\_Card\Lab\_02\_Credit\_Card\_Fraud\solution



## Appendix A: Python Script for PDI

---

```
# the following libraries need to be installed (see pre-requisites)
# pandas
# matplotlib
# py4j
# numpy
# TPOT

# import required libraries
import pandas as pd
import numpy as np

from tpot import TPOTClassifier
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

# the dataset is referenced in the step: sv-change_to_numbers
# independent variables (x) are referenced starting at row 1: col 1. -1
# references all columns apart from the last
x = dataset.iloc[:,1:-1].values

# transform features by scaling each feature to a given range
min_max_scaler = preprocessing.MinMaxScaler()

# compute the data minimum and maximum for scaling, then transform.
x_scaled = min_max_scaler.fit_transform(x)

# optional - change to numpy array
X=np.asarray(x_scaled)
y=np.asarray(dataset.iloc[:,-1])

# split the dataset into train and test. Test size is set at 75% of dataset
# (10,000 rows)
# further details on random_state:
#
https://het.as.utexas.edu/HET/Software/Numpy/reference/generated/numpy.random.RandomState.html
```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.75,
random_state=None)

# set TPOT parameters (see Appendix: B for further details)
tpot = TPOTClassifier(generations=1, verbosity=2, population_size=100,
scoring='accuracy', config_dict='TPOT light')

tpot.fit(X_train, y_train)

output_score=str(tpot.score(X_test, y_test))

# export TPOT results as python script
tpot.export('tpot_creditcard_pipeline.py')

# print the TPOT result
print(tpot.score(X_test, y_test))

# PDI output fields which are defined in a dataframe which is mapped to a
PDI output field: model_df
model_name=[x[0] for x in tpot.evaluated_individuals_.items()]
model_gen=[x[1]['generation'] for x in tpot.evaluated_individuals_.items()]
model_mut=[x[1]['mutation_count'] for x in
tpot.evaluated_individuals_.items()]
model_cross=[x[1]['crossover_count'] for x in
tpot.evaluated_individuals_.items()]
model_predec=[x[1]['predecessor'] for x in
tpot.evaluated_individuals_.items()]
model_opp=[x[1]['operator_count'] for x in
tpot.evaluated_individuals_.items()]
model_cv=[str(y[1]['internal_cv_score']) for y in
tpot.evaluated_individuals_.items()]
model_list=list(zip(model_name,model_gen,model_mut,model_cross,model_predec
,model_opp,model_cv))
model_df=pd.DataFrame(model_list,columns=['pipe','generation','mutation','c
rossover','predecessor','operator','cv'])

```

---

## Appendix B: TPOT Parameters

Parameter	Valid values	Effect
<b>generation</b>	Any positive integer	The number of generations to run pipeline optimization over. Generally, TPOT will work better when you give it more generations (and therefore time) to optimize over. TPOT will evaluate <code>generations x population_size</code> number of pipelines in total.
<b>population_size</b>	Any positive integer	The number of individuals in the GP population. Generally, TPOT will work better when you give it more generations (and therefore time) to optimize over. TPOT will evaluate <code>generations x population_size</code> number of pipelines in total.
<b>mutation_rate</b>	[0.0, 1.0]	The mutation rate for the genetic programming algorithm in the range [0.0, 1.0]. This tells the genetic programming algorithm how many pipelines to apply random changes to every generation. We don't recommend that you tweak this parameter unless you know what you're doing.
<b>crossover_rate</b>	[0.0, 1.0]	The crossover rate for the genetic programming algorithm in the range [0.0, 1.0]. This tells the genetic programming algorithm how many pipelines to "breed" every generation. We don't recommend that you tweak this parameter unless you know what you're doing.
<b>num_cv_folds</b>	[2, 10]	The number of folds to evaluate each pipeline over in k-fold cross-validation during the TPOT pipeline-optimization process.

Parameter	Valid values	Effect
<b>scoring</b>	'accuracy', 'adjusted_rand_score', 'average_precision', 'f1', 'f1_macro', 'f1_micro', 'f1_samples', 'f1_weighted', 'log_loss', 'mean_absolute_error', 'mean_squared_error', 'median_absolute_error', 'precision', 'precision_macro', 'precision_micro', 'precision_samples', 'precision_weighted', 'r2', 'recall', 'recall_macro', 'recall_micro', 'recall_samples', 'recall_weighted', 'roc_auc' or a callable function with signature <code>scorer(y_true, y_pred)</code>	Function used to evaluate the quality of a given pipeline for the problem. By default, balanced accuracy is used for classification and mean squared error is used for regression. TPOT assumes that any function with "error" or "loss" in the name is meant to be minimized, whereas any other functions will be maximized.
<b>max_time_mins</b>	Any positive integer	How many minutes TPOT has to optimize the pipeline. This setting will override the <code>generations</code> parameter.
<b>random_state</b>	Any positive integer	The random number generator seed for TPOT. Use this to make sure that TPOT will give you the same results each time you run it against the same dataset with that seed.
<b>verbosity</b>	{0, 1, 2, 3}	How much information TPOT communicates while it's running. 0 = none, 1 = minimal, 2 = high, 3 = all. A setting of 2 or higher will add a progress bar to calls to <code>fit()</code> .
<b>disable_update_check</b>	[True, False]	Flag indicating whether the TPOT version checker should be disabled.



Further details can be found at [Using TPOT](#).

## Appendix C: R Script for Train

---

```
# the following packages need to be installed (see pre-requisites)
# rJava
# randomForest

# load randomForest package
library(randomForest)

# the dataset is referenced in the step: sv-convert_booleans_to_numbers
# assign the dataframe: train to variable: train.df
train.df <- as.data.frame(train)

# train model with 8 trees takes about 50 seconds on this VM
# declare variable rf assigned the results
# ( y) dependent variable = train.df (dataset) reported_as_fraud_historic
# (column)
# ~ (tilde) . (point) = (x) independent variables
rf <- randomForest(train.df$reported_as_fraud_historic ~ ., train.df,
ntree=8, importance=TRUE)

# save model to output folder: Note no spaces and double backslashes are
# required.
save(rf, file="C:\\Machine--
Learning\\01_Credit_Card\\Lab_02_Credit_Card_Fraud\\train_model_output\\rf.
rdata")

# print message ok to indicate no probs..
# declare variable: ok assigned with the value "Finished"
ok <- "Finished"

# assign variable ok to dataframe: ok.df
ok.df <- as.data.frame(ok)
ok.df
```

---

## Appendix D: R Script for Predict

---

```
# the following packages need to be installed (see pre-requisites)
# rJava
# randomForest

# load randomForest package
library(randomForest)

# the dataset is referenced in the step: sv-convert_booleans_to_numbers
# assign the dataframe: test to variable: test.df
test.df <- as.data.frame(test)

# load randomForest Model
load("C:\\Machine--
Learning\\01_Credit_Card\\Lab_02_Credit_Card_Fraud\\train_model_output\\rf.
rdata")

# declare a variable pred which is assigned the results of using the
predict function
# predict function runs our randomForest model against test.df = newdata
# predict with new test data
pred <- predict(rf, newdata = test.df)

# assign pred dataframe to pred.df variable
pred.df <- as.data.frame(pred)

# prepare output data
# using cbind function to bind pred.df to test.df
submission <- data.frame(cbind(test.df,pred.df))

# output data
submission
```

---

## Related Information

Here are some links to information that you may find helpful while using this lab guide:

- [Google Colab](#)
- [Pentaho Data Integration Best Practices Library](#)
- [Python Executor](#)
- [Python for Windows](#)
- [r-project](#)
- [RStudio IDE](#)