**Data Analytics Coursework**

Steps taken to clean data.

Replace age text with numerical equivalent.

Replace ages less than minimum credit application age and maximum living age with the medium age value as an imputation.

# Check if the age is less than 18 or more than 122, and if so, replace it with 33, otherwise leave it as is.

if cells['Applicant Age'].value < 18 or cells['Applicant Age'].value > 122:

return 33

else:

return cells['Applicant Age'].value

Binned Savings values as none-observed, low, moderate, moderate-high, High

Binned employment term as N/A for unemployed, short, medium, medium-long and long

Open refine automatically Converted credit amount to integer thus removing the integer separating commas.

**Age binning to get the nominal datatype.**

# Jython script for OpenRefine to categorize age

# Ensure the age is a number; if not, return "Invalid Age"

try:

age = int(value)

except ValueError:

return "Invalid Age"

# Categorise age into respective bins: Young Adult, Adult, Senior

if age < 30:

return "Young Adult"

elif 30 <= age < 50:

return "Adult"

else:

return "Senior"

**Binning Credit amount:**

# Ensure the credit amount is a number; if not, return "Invalid Amount"

try:

amount = int(value)

except ValueError:

return "Invalid Amount"

# Categorize credit amount into respective bins: Low, Medium, High, Very High

if amount <= 1500:

return "Low"

elif 1501 <= amount <= 5000:

return "Moderate"

elif 5001 <= amount <= 10000:

return "High"

else:

return "Very High"

**Savings nominal binning**

# Example Jython script for OpenRefine to re-categorize 'Savings' column

value = getValue('Savings').value

# get the current value in 'Savings' column

# Define mapping from original categories to new categories

savings\_mapping = {

'none-found': ‘unknown’,

'moderate-high': ‘high’,

'high': 'very\_high’,

'moderate': 'moderate'

}

# Return the new category

return savings\_mapping.get(value, 'Unknown') # default to 'Unknown' for any unforeseen values