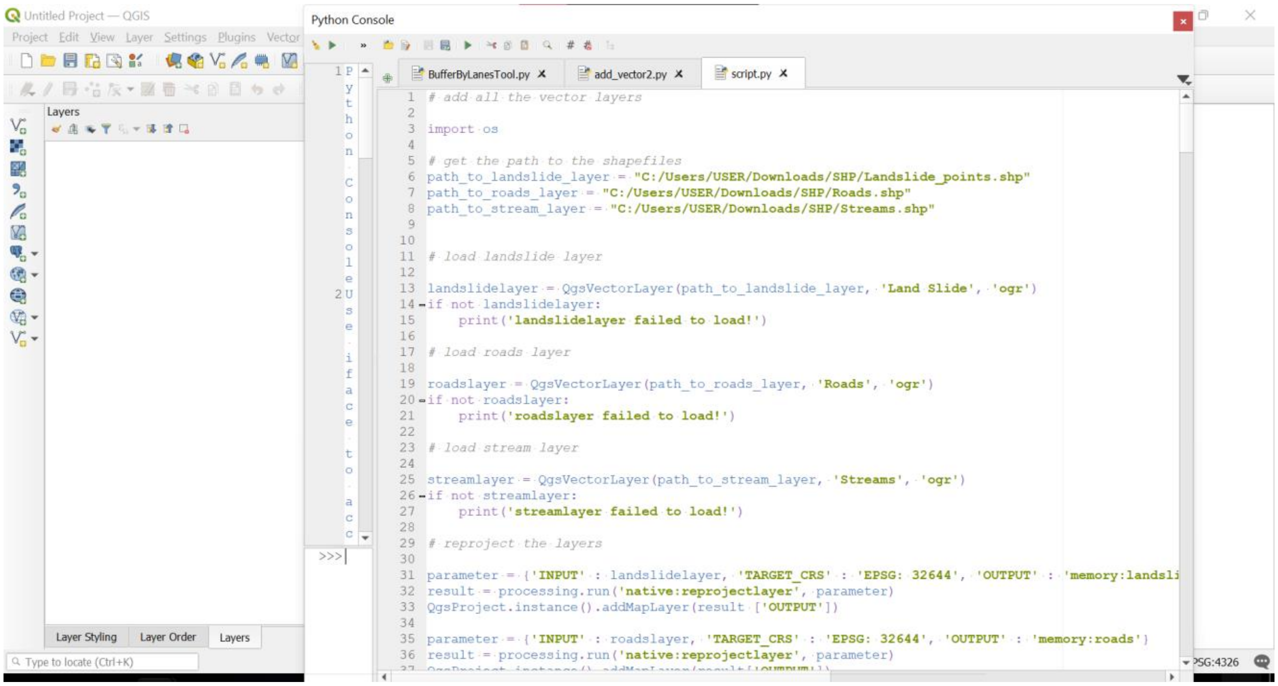
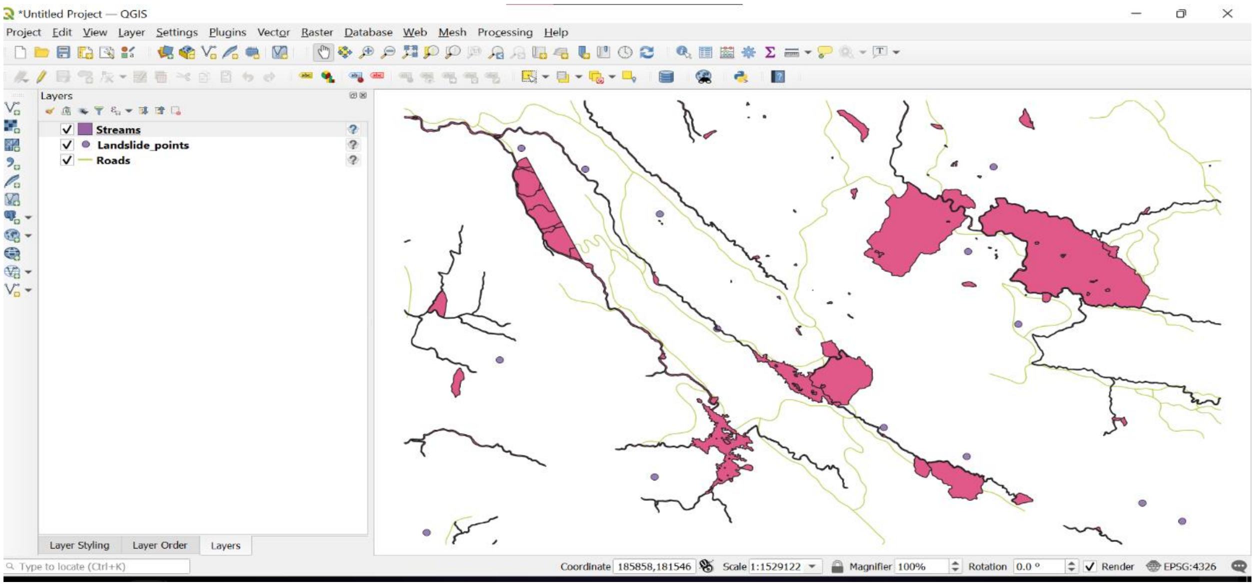
APPLICATION OF SPATIAL DATA TO IDENTIFY ROAD RISKSIN NUWARA-ELIYA, AMBAGAMUWA AREA, SRI LANKA

Screenshots of the processes are as follows:

i. Import the data into QGIS environment using python snippet as shown below



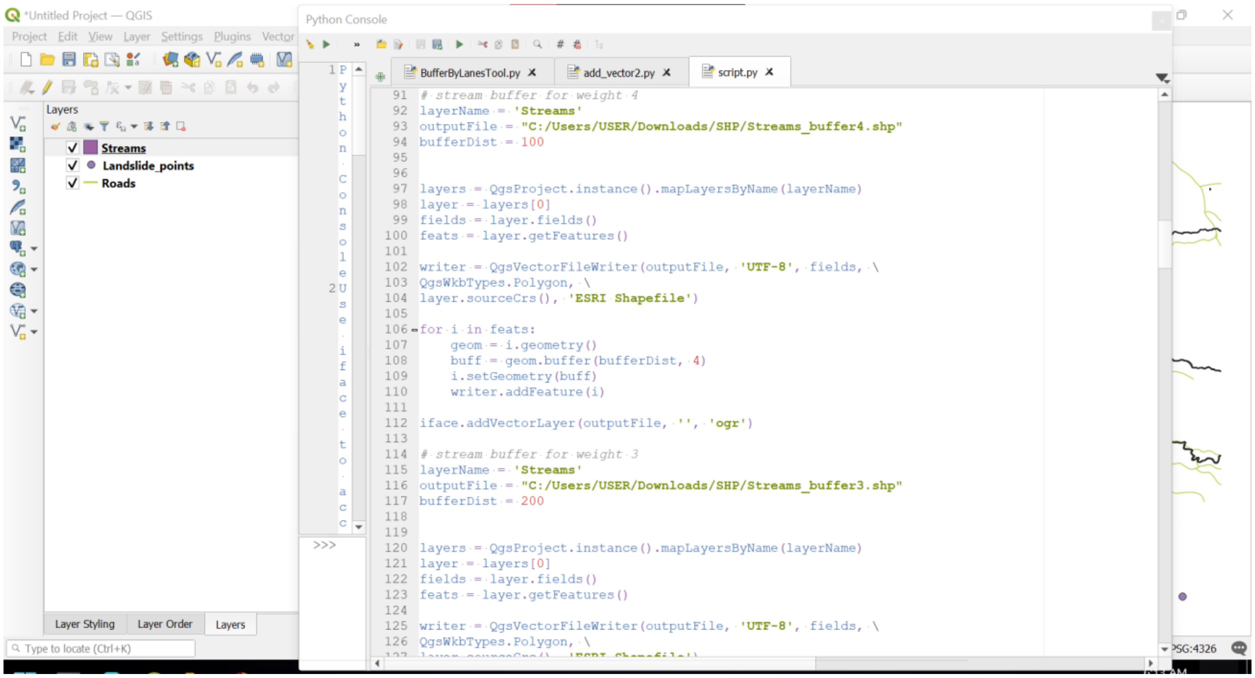
```
# add all the vector layers

import os

# get the path to the shapefiles
path_to_landslide_layer = "C:/Users/USER/Downloads/SHP/Landslide_points.shp"
path_to_roads_layer = "C:/Users/USER/Downloads/SHP/Roads.shp"
path_to_stream_layer = "C:/Users/USER/Downloads/SHP/Streams.shp"


# load landslide layer

landslidelayer = QgsVectorLayer(path_to_landslide_layer, 'Land Slide', 'ogr')
if not landslidelayer:
    print('landslidelayer failed to load!')

# load roads layer

roadslayer = QgsVectorLayer(path_to_roads_layer, 'Roads', 'ogr')
if not roadslayer:
    print('roadslayer failed to load!')

# load stream layer

streamlayer = QgsVectorLayer(path_to_stream_layer, 'Streams', 'ogr')
if not streamlayer:
    print('streamlayer failed to load!')

# reproject the layers

parameter = {'INPUT' : landslidelayer, 'TARGET_CRS' : 'EPSG: 32644', 'OUTPUT' : 'memory:landsli
result = processing.run('native:reprojectlayer', parameter)
QgsProject.instance().addMapLayer(result ['OUTPUT'])

parameter = {'INPUT' : roadslayer, 'TARGET_CRS' : 'EPSG: 32644', 'OUTPUT' : 'memory:roads'}
result = processing.run('native:reprojectlayer', parameter)
```



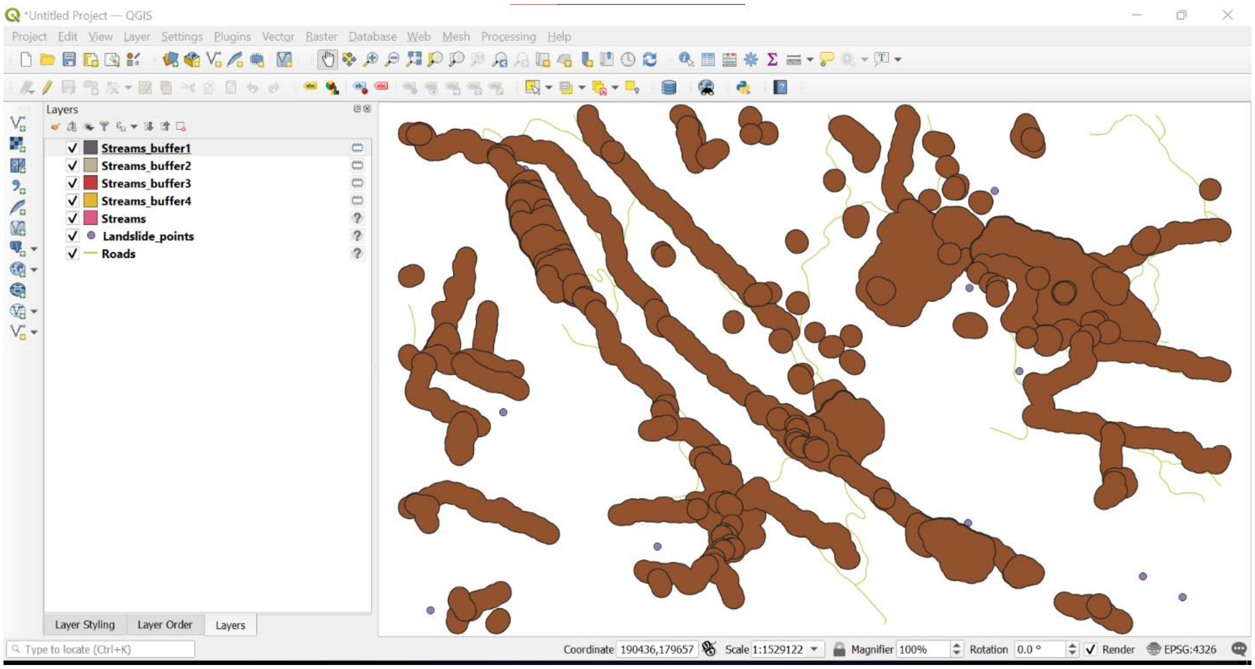
Coordinate 185858,181546 Scale 1:1529122 Magnifier 100% Rotation 0.0 ° ☑ Render EPSG:4326

The result the algorithm when run.

ii. The next procedure is to buffer the Landslide points and stream. The algorithm used is shown below with the resulting buffer



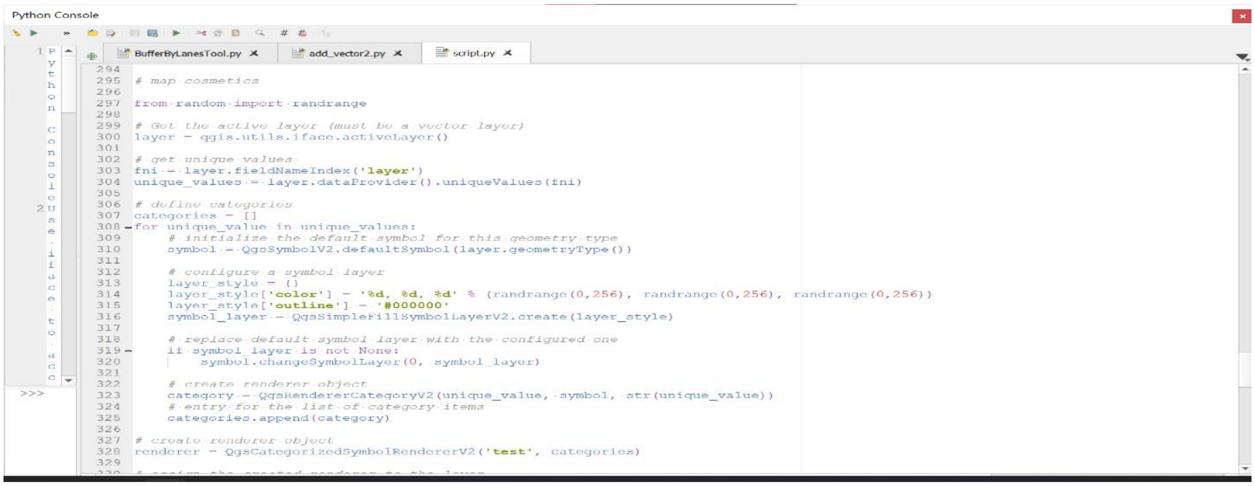
```
# stream buffer for weight 4
layerName = 'Streams'
outputFile = "C:/Users/USER/Downloads/SHP/Streams_buffer4.shp"
bufferDist = 100


layers = QgsProject.instance().mapLayersByName(layerName)
layer = layers[0]
fields = layer.fields()
feats = layer.getFeatures()

writer = QgsVectorFileWriter(outputFile, 'UTF-8', fields, \
QgsWkbTypes.Polygon, \
layer.sourceCrs(), 'ESRI Shapefile')

for i in feats:
    geom = i.geometry()
    buff = geom.buffer(bufferDist, 4)
    i.setGeometry(buff)
    writer.addFeature(i)

iface.addVectorLayer(outputFile, '', 'ogr')

# stream buffer for weight 3
layerName = 'Streams'
outputFile = "C:/Users/USER/Downloads/SHP/Streams_buffer3.shp"
bufferDist = 200


layers = QgsProject.instance().mapLayersByName(layerName)
layer = layers[0]
fields = layer.fields()
feats = layer.getFeatures()

writer = QgsVectorFileWriter(outputFile, 'UTF-8', fields, \
QgsWkbTypes.Polygon, \
layer.sourceCrs(), 'ESRI Shapefile')
```



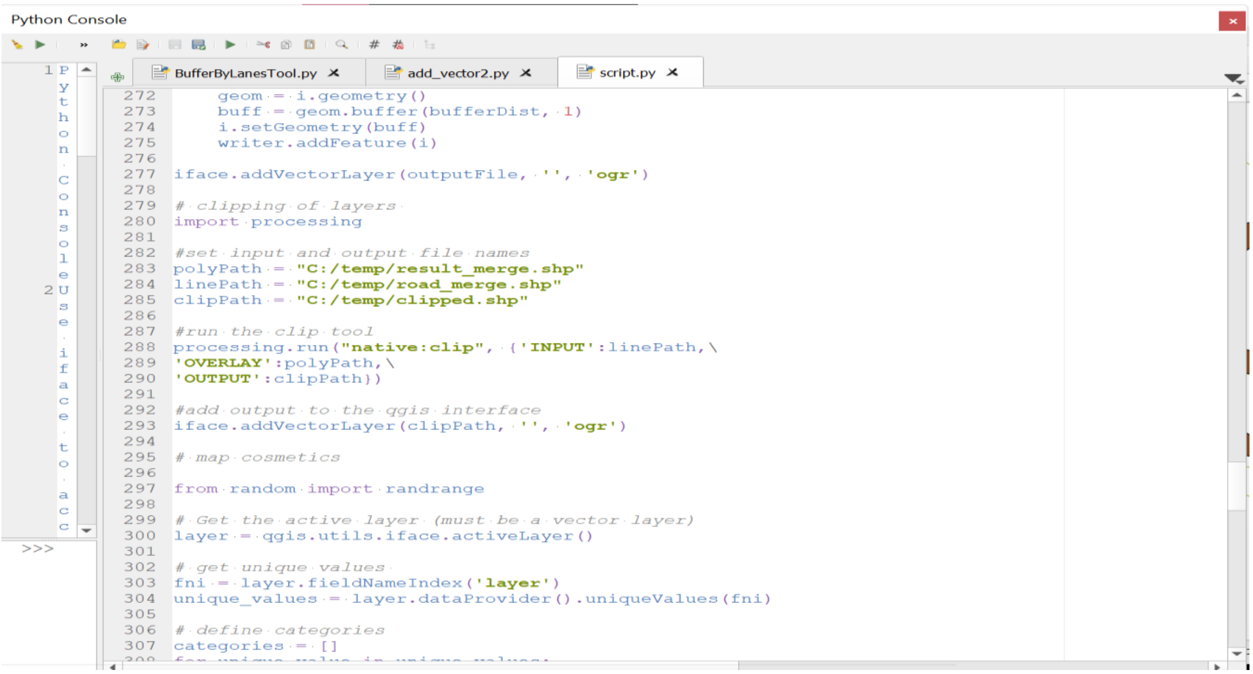
iii. The next step is to clip the corresponding weight assigned to each layer. The algorithm is as shown below.

iv. The next step is map cosmetics, in order to design the cartographically.

The code snippet is shown below



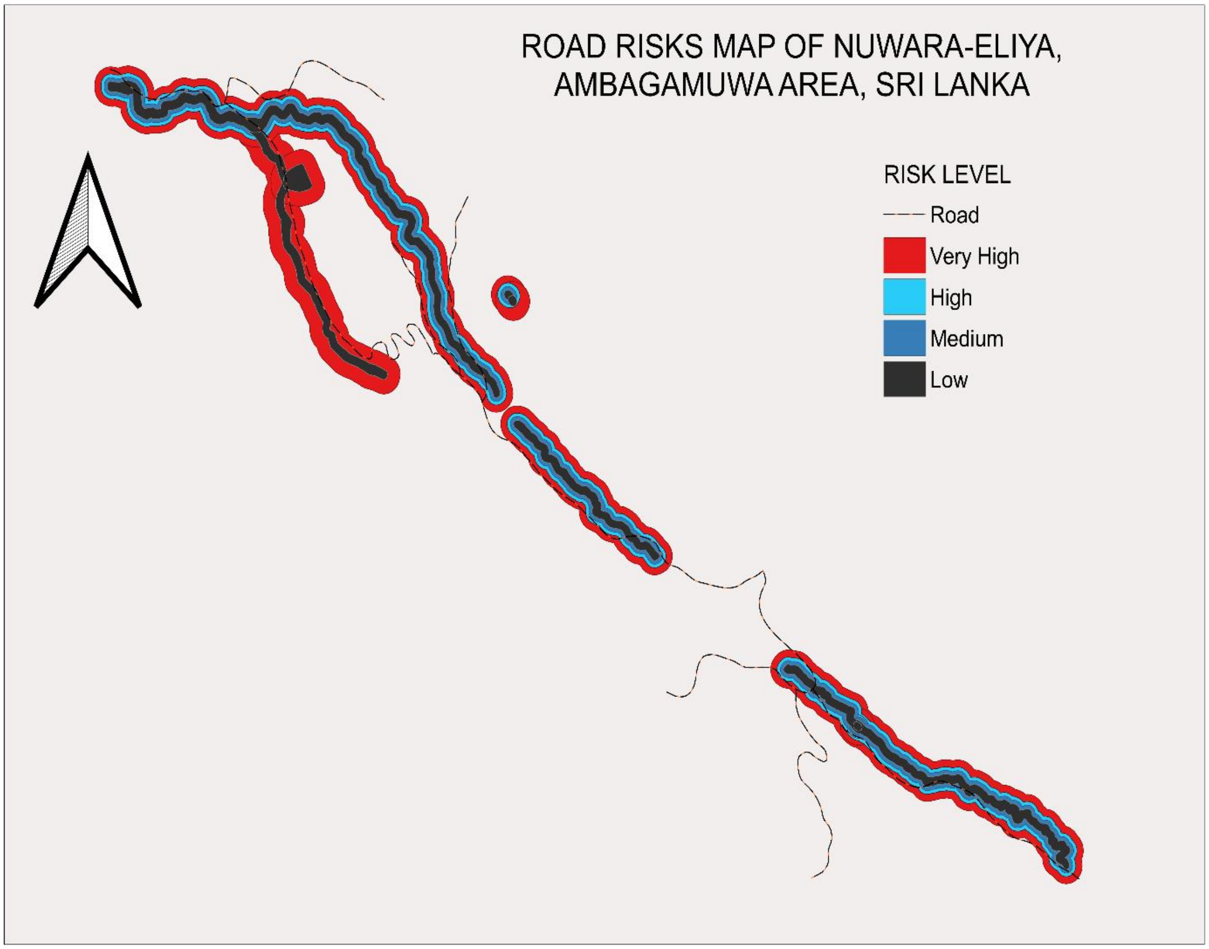
```
# map cosmetics

from random import randrange

# Get the active layer (must be a vector layer)
layer = qgis.utils.iface.activeLayer()

# get unique values
fni = layer.fieldNameIndex('layer')
unique_values = layer.dataProvider().uniqueValues(fni)

# define categories
categories = []
for unique_value in unique_values:
    # initialize the default symbol for this geometry type
    symbol = QgsSymbolV2.defaultSymbol(layer.geometryType())

    # configure a symbol layer
    layer_style = {}
    layer_style['color'] = '%d, %d, %d' % (randrange(0,256), randrange(0,256), randrange(0,256))
    layer_style['outline'] = '#000000'
    symbol_layer = QgsSimpleFillSymbolLayerV2.create(layer_style)

    # replace default symbol layer with the configured one
    if symbol_layer is not None:
        symbol.changeSymbolLayer(0, symbol_layer)

    # create renderer object
    category = QgsRendererCategoryV2(unique_value, symbol, str(unique_value))
    # entry for the list of category items
    categories.append(category)

# create renderer object
renderer = QgsCategorizedSymbolRendererV2('test', categories)

```



```
Python Console
Use iface to acc
>>>
```

```python
    geom = i.geometry()
    buff = geom.buffer(bufferDist, 1)
    i.setGeometry(buff)
    writer.addFeature(i)

iface.addVectorLayer(outputFile, '', 'ogr')

# clipping of layers
import processing

#set input and output file names
polyPath = "C:/temp/result_merge.shp"
linePath = "C:/temp/road_merge.shp"
clipPath = "C:/temp/clipped.shp"

#run the clip tool
processing.run("native:clip", {'INPUT':linePath,\
'OVERLAY':polyPath,\
'OUTPUT':clipPath})

#add output to the qgis interface
iface.addVectorLayer(clipPath, '', 'ogr')

# map cosmetics

from random import randrange

# Get the active layer (must be a vector layer)
layer = qgis.utils.iface.activeLayer()

# get unique values
fni = layer.fieldNameIndex('layer')
unique_values = layer.dataProvider().uniqueValues(fni)

# define categories
categories = []
for unique_value in unique_values:
```

After running the code, the map generated is shown below:


ROAD RISKS MAP OF NUWARA-ELIYA, AMBAGAMUWA AREA, SRI LANKA
RISK LEVEL
Road
Very High
High
Medium
Low