session4_pyspark_solution

September 23, 2019

1 [MIRACUM 2019][Session 4] Solution

- Kim Hee (Graduate research assistant)
- Universitätsmedizin Mannheim, Mannheim (UMM)

- This is prepared for a turorial `Data analysis tools (Datenanalysewerkzeuge)`

1.0.1 Protocol:

1. Create a Spark Context 2. Import required libraries 3. Load the cholera death data 4. Create a KMeans model 5. Assign centroids to the data 6. Visualize the result

[1]:
```
from pyspark import SparkContext
sc = SparkContext()
sc
```

[1]:
```
<SparkContext master=local[*] appName=pyspark-shell>
```

1.0.2 Protocol:

1. Create a Spark Context 2. Import required libraries 3. Load the cholera death data 4. Create a KMeans model 5. Assign centroids to the data 6. Visualize the result

[2]:
```
import pyspark.mllib.clustering
from pyspark.mllib.clustering import KMeans, KMeansModel
import pandas as pd
import numpy as np
import folium
from folium import plugins
import matplotlib.pyplot as plt
%matplotlib inline
```

1.0.3 Protocol:

1. Create a Spark Context 2. Import required libraries 3. Load the cholera death data 4. Create a KMeans model 5. Assign centroids to the data 6. Visualize the result

[3]:
```
raw = sc.textFile("data/cholera_deaths.csv")
header = sc.parallelize([raw.first()])
raw = raw.subtract(header)
raw.take(2)
```

[3]:
```
['1,2,-0.137883,51.513361', '13,1,-0.138378,51.513169']
```

[4]:
```
line = raw.map(lambda x: x.split(','))
line.take(2)
```

[4]:
```
[['1', '2', '-0.137883', '51.513361'], ['13', '1', '-0.138378', '51.513169']]
```

[5]:
```
geo = line.map(lambda x: (float(x[2]), float(x[3])))
geo.take(2)
```

[5]:
```
[(-0.137883, 51.513361), (-0.138378, 51.513169)]
```

1.0.4 Protocol:

1. Create a Spark Context 2. Import required libraries 3. Load the cholera death data 4. Create a KMeans model with `k=3` 5. Assign centroids to the data 6. Visualize the result

[6]:
```
k = 3
model = KMeans.train(geo, k)
```

1.0.5 Protocol:

1. Create a Spark Context 2. Import required libraries 3. Load the cholera death data 4. Create a KMeans model 5. Assign centroids to the data 6. Visualize the result

[7]:
```
clusters = model.predict(geo)
clusters.take(20)
```

[7]:
```
[2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
```

1.0.6 Protocol:

1. Create a Spark Context 2. Import required libraries 3. Load the cholera death data 4. Create a KMeans model 5. Assign centroids to the data 6. Visualize the result

[8]:
```
# data preparatoin for VIZ
lat = geo.map(lambda x: float(x[0])).collect()
lon = geo.map(lambda x: float(x[1])).collect()
clusters = clusters.collect()
```

```
df_list = list(map(lambda x,y,z:[x,y,z], lon,lat,clusters))
df_list[:5]
```

[8]:
```
[[51.513361, -0.137883, 2],
 [51.513169, -0.138378, 2],
 [51.513402, -0.139045, 2],
 [51.513676, -0.138887, 2],
 [51.512885, -0.138624, 2]]
```

[9]:
```
# the latitude and Longitude coordinates of the Soho District center
SOHO_COORDINATES = (51.513578, -0.136722)
map_soho = folium.Map(location = SOHO_COORDINATES, width = "100%", zoom_start =␣
↪15) # max zoom: 18
folium.TileLayer('cartodbpositron').add_to(map_soho) # stamentoner
map_soho
```

[9]:
```
<folium.folium.Map at 0x7f6cc21590f0>
```

[10]:
```
def my_palettes(cluster):
    if cluster ==  0:
        return 'green'
    elif cluster ==  1:
        return 'yellow'
    elif cluster ==  2:
        return 'red'
    elif cluster ==  3:
        return 'blue'
    elif cluster ==  4:
        return 'orange'
    elif cluster ==  5:
        return 'skyblue'
    elif cluster ==  6:
        return 'gray'
    elif cluster ==  7:
        return 'purple'
    else:
        return 'black'
```

[11]:
```
for i in range(0, len(df_list)):
    folium.RegularPolygonMarker(location = df_list[i][:2], \
                                stroke = False, \
                                fill_color = my_palettes(df_list[i][2]), \
                                fill_opacity = 0.5, number_of_sides = 12, ␣
↪radius = 3
                               ).add_to(map_soho)
map_soho
```

[11]:
```
<folium.folium.Map at 0x7f6cc21590f0>
```

[ ]: