Capstone_Applied_DS

June 10, 2019

1 Coursera Data Science - Capstone Project - Final

In [1]:
```
import pandas    as pd
import numpy     as np
import urllib.request  as request

import json                               # library to handle JSON files
import matplotlib.cm      as cm           # Matplotlib plotting modules
import matplotlib.colors as colors
import folium
from folium.features import DivIcon
import matplotlib         as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import re

from sklearn.preprocessing import StandardScaler
from geopy.geocoders       import Nominatim      # convert an address into latitude and longitude values
from pandas.io.json        import json_normalize # tranform JSON file into a pandas dataframe
from sklearn.cluster       import KMeans         # import k-means from clustering stage
from os import path
```

1.1 Get Source Data

1.1.1 The file is store in GIT hub

You can find more Australia Statistic Data from the following URL:

https://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/1410.02013-18?OpenDocument

In [2]:
```
# Load OZ_Population.xls (Population and People, ASGS, 2011 to 2018 )

url_src  = 'https://www.abs.gov.au/AUSSTATS/subscriber.nsf/log?openagent&14100ds0001_2011-18.xls&
file_src = 'OZ_Population.xls'

if not path.exists( file_src ):
   print('Please wait, downloading {} from ABS .....'.format(file_src))
```

```
        request.urlretrieve(url_src, file_src)

    print('Loading {} .....'.format(file_src))
    data_src1 = pd.read_excel(url_src, skiprows=7)
    print('Load {} is complete with {} rows!\n'.format(file_src, data_src1.shape[0]))
```

Loading OZ_Population.xls ...
Load OZ_Population.xls is complete with 21679 rows!

```
In [3]: # Load OZ_Edu_Empl.xls (Education and Employment, ASGS, 2011 to 2018)

    url_src  = 'https://www.abs.gov.au/AUSSTATS/subscriber.nsf/log?openagent&14100ds0007_2011-18.xls&
    file_src = 'OZ_Edu_Empl.xls'

    if not path.exists( file_src ):
        print('Please wait, downloading {} from ABS .....'.format(file_src))
        request.urlretrieve(url_src, file_src)

    print('Loading {} .....'.format(file_src))
    data_src2 = pd.read_excel(file_src,skiprows=6)
    print('Load {} is complete with {} rows!\n'.format(file_src, data_src2.shape[0]))
    data_src2.tail(9).tail(2)
```

Loading OZ_Edu_Empl.xls ...
Load OZ_Edu_Empl.xls is complete with 18803 rows!

```
Out[3]:                        Unnamed: 0 Unnamed: 1 Unnamed: 2  \
    18801                              NaN        NaN        NaN
    18802  Ï Commonwealth of Australia 2019        NaN        NaN

         4 year olds enrolled in preschool or in a preschool program  \
    18801                                                NaN
    18802                                                NaN

         5 year olds enrolled in preschool or in a preschool program  \
    18801                                                NaN
    18802                                                NaN

         Enrolled in preschool  \
    18801                  NaN
    18802                  NaN

         Enrolled in a preschool program within a long day care centre  \
    18801                                                NaN
    18802                                                NaN
```

```
      Children enrolled  across more than one provider type  \
18801                                                    NaN
18802                                                    NaN

      Total enrolled in a preschool program Less than 15 hours  ...  \
18801                                       NaN                 NaN  ...
18802                                       NaN                 NaN  ...

      Number of Employee Jobs - Public administration and safety  \
18801                                                    NaN
18802                                                    NaN

      Number of Employee Jobs - Education and training  \
18801                                                  NaN
18802                                                  NaN

      Number of Employee Jobs - Health care and social assistance  \
18801                                                    NaN
18802                                                    NaN

      Number of Employee Jobs - Arts and recreation services  \
18801                                                    NaN
18802                                                    NaN

      Number of Employee Jobs - Other services  \
18801                                        NaN
18802                                        NaN

      Number of Employee Jobs - Total Labour Force Unemployed  \
18801                                 NaN          NaN        NaN
18802                                 NaN          NaN        NaN

      Unemployment rate  Participation rate
18801               NaN                 NaN
18802               NaN                 NaN

[2 rows x 83 columns]
```

1.2 Data Cleansing for OZ_Population.xls

In [4]:
```
Col_Nbrs  = list(range(0,3)) + [72] + list(range(85, 106))
Col_Names = ['CODE','Suburb','YEAR','Median Age','Born_in_Oceania_Ex_OZ','Born_in_North_We

data1 = data_src1.iloc[:, Col_Nbrs]
data1.columns = Col_Names
```

In [5]:
```
data_src1.tail(10).head(3)
```

```
Out[5]:     Unnamed: 0     Unnamed: 1 Unnamed: 2 0-14 years 15-24 years  \
21669  901031003     Jervis Bay       2018          -           -
21670  901041004  Norfolk Island       2016       17.2         6.3
21671  901041004  Norfolk Island       2017       16.7         7.4

      25-34 years 35-44 years 45-54 years 55-64 years 65-74 years  ...  \
21669           -           -           -           -           -  ...
21670         6.7          13        15.4        17.6        14.2  ...
21671         6.2        12.2        14.8        18.3          15  ...

      Unnamed: 97 Unnamed: 98 Unnamed: 99 Unnamed: 100 Unnamed: 101  \
21669           -           -           -            -            -
21670         0.3           -           -          0.4         27.3
21671           -           -           -            -            -

      Unnamed: 102 Unnamed: 103 Unnamed: 104 Unnamed: 105 Unnamed: 106
21669            -            -            -            -            -
21670          9.5         81.5         11.6          6.9         48.7
21671            -            -            -            -            -

[3 rows x 107 columns]
```

```
In [6]: code_num  = data1.CODE.apply(pd.to_numeric, errors='coerce')
        code_filt = (code_num > 200000000) & (code_num < 300000000)   # Melbourne Data
        data_pop  = data1[code_filt].reset_index(drop=True)
        print(data_pop.shape)
        data_pop.tail(11).head(2)
```

```
(3665, 25)
```

```
Out[6]:          CODE              Suburb  YEAR Median Age Born_in_Oceania_Ex_OZ  \
3654  217041479  Warrnambool - North  2016         38                   1.3
3655  217041479  Warrnambool - North  2017       38.1                     -

     Born_in_North_West_Europe Born_in_Southern_Eastern_Europe  \
3654                       2.7                             0.4
3655                         -                               -

     Born_in_North_Africa_Middle_East Born_in_South_East_Asia  \
3654                              0.2                     0.7
3655                                -                       -

     Born_in_North_East_Asia  ...  Relg_Christianity Relg_Hinduism Relg_Islam  \
3654                     1.1  ...               57.5           0.3        0.2
3655                       -  ...                  -             -          -

     Relg_Judaism Relg_Other Relg_Secular Relg_not_stated Residency_Citizen  \
```

```
3654          -      0.2       31.5          9.6            89.4
3655          -        -          -            -              -

     Residency_not_Citizen Residency_not_Stated
3654                   3.9                  6.7
3655                     -                    -

[2 rows x 25 columns]
```

1.3 Data Cleansing for OZ_Edu_Empl.xls

In [10]:
```
Col_Names = ['CODE','YEAR','Edu_PostGraduate','Edu_Diploma','Edu_Bachelor','Edu_Adv_Dipl','
Col_Nbrs  = [data_src2.columns[0],data_src2.columns[2],'Postgraduate Degree','Graduate Diploma, Grad

data2 = data_src2[Col_Nbrs]
data2.columns = Col_Names
data2.shape
```

Out[10]: (18803, 7)

In [11]:
```
code_num  = data2.CODE.apply(pd.to_numeric, errors='coerce')
code_filt = (code_num > 20000000) & (code_num < 300000000)   # Melbourne Data
data_edu  = data2[code_filt].reset_index(drop=True)
print(data_edu.shape)
data_edu.head(3)
```

```
(7184, 7)
```

Out[11]:
```
        CODE  YEAR Edu_PostGraduate Edu_Diploma Edu_Bachelor Edu_Adv_Dipl  \
0  101021007  2011              4.4         2.7         12.3          7.7
1  101021007  2012                -           -            -            -
2  101021007  2013                -           -            -            -

  Unempl_Rate
0           4
1           -
2           -
```

1.4 Merge OZ_Population.xls and OZ_Edu_Empl.xls into a single table

1.4.1 Use Year 2016 as it has the most complete data

In [12]:
```
data_merged_tmp = data_pop.join(data_edu.set_index(['CODE','YEAR']), on=['CODE','YEAR'], how

data = data_merged_tmp[ data_merged_tmp['YEAR'] == 2016]
data.reset_index(drop=True,inplace=True)
```

In [13]:
```
#### Replace all missing values with Zeroes #####
data.iloc[:,3:].replace(to_replace = '-', value = 0, inplace=True)
```

```
data.drop( 'Born_Overseas', axis = 1)

print('Number of Rows After Merge  :', data_merged_tmp.shape[0])
print('Number of Rows For Year 2016:', data.shape[0])

data[['Relg_Judaism']].head()
```

```
Number of Rows After Merge  : 3665
Number of Rows For Year 2016: 459


/home/jupyterlab/conda/lib/python3.6/site-packages/pandas/core/frame.py:4042: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-vie
  method=method)
```

Out[13]:

	Relg_Judaism
0	0.1
1	0.1
2	0.1
3	0
4	0

In [14]:
```
data.columns[3:]
```

Out[14]:
```
Index(['Median Age', 'Born_in_Oceania_Ex_OZ', 'Born_in_North_West_Europe',
       'Born_in_Southern_Eastern_Europe', 'Born_in_North_Africa_Middle_East',
       'Born_in_South_East_Asia', 'Born_in_North_East_Asia',
       'Born_in_Southern_Central_Asia', 'Born_in_America',
       'Born_in_Sub_Saharan_Africa', 'Born_Overseas', 'Relg_Buddhism',
       'Relg_Christianity', 'Relg_Hinduism', 'Relg_Islam', 'Relg_Judaism',
       'Relg_Other', 'Relg_Secular', 'Relg_not_stated', 'Residency_Citizen',
       'Residency_not_Citizen', 'Residency_not_Stated', 'Edu_PostGraduate',
       'Edu_Diploma', 'Edu_Bachelor', 'Edu_Adv_Dipl', 'Unempl_Rate'],
      dtype='object')
```

1.5 Get Latitude and Longitude of all suburbs

In [15]:
```
file_src = 'GeoLoc.csv'
geolocator = Nominatim(user_agent="Capstone")

#######
### Get Geo Location if the GeoLoc.csv does not exist ###
#######
if not path.exists( file_src ):
    print('Please wait, getting Geo Location from Nominatim .....')
```

```
    GeoLoc = {}
    for suburb in data['Suburb']:
        sub_vic = suburb + ", Vic"
        loc = geolocator.geocode(sub_vic)
        if loc is not None:
            GeoLoc[suburb] = [loc.latitude, loc.longitude]

    GeoLoc_dict = pd.DataFrame.from_dict(GeoLoc, orient='index')
    GeoLoc_dict.columns = ['latitude', 'longitude']
    GeoLoc_dict.to_csv('GeoLoc.csv')

GeoLoc = pd.read_csv(file_src, index_col=0)
GeoLoc.head()

cbd_loc = geolocator.geocode("City of Melbourne, Vic")
```

1.6 Define a Function to Get Columns based on the interest

In [17]:
```
def GetColumns ( p_interest ):

    cols = []
    for col in data.columns:
        if re.match(p_interest + '.*', col) :
            cols = cols + [col]
    return cols
```

1.7 Do Clustering based on Religion based

In [18]:
```
#cols = GetColumns('Relg')
cols = ['Relg_Christianity', 'Relg_Buddhism', 'Relg_Hinduism', 'Relg_Islam', 'Relg_Judaism']
```

1.7.1 Normalized the data

In [19]:
```
from sklearn.preprocessing import scale

religion_df = pd.DataFrame()

for i in range( len(cols) ):
    col_nm = cols[i]
    religion_df[ col_nm ] = scale( data[ col_nm ].astype(float) )

    v_min   = religion_df[ col_nm ].min()
    v_range = religion_df[ col_nm ].max() - v_min
    religion_df[ col_nm ] = ( religion_df[ col_nm ] - v_min ) / v_range
```

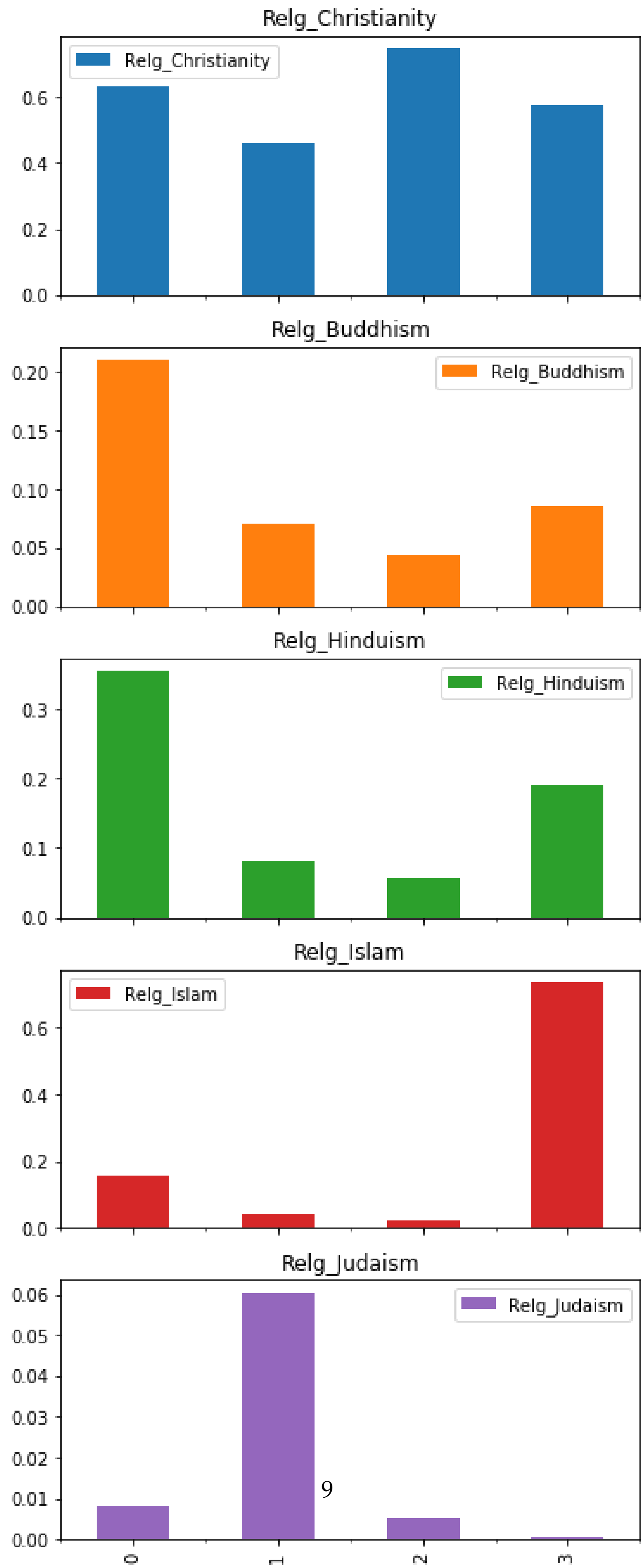
1.7.2 Define Procedure to do Clustering

In [20]:
```
def Clustering_n ( p_clus, data_df ):

    clust_k_means = KMeans(init = "k-means++", n_clusters = p_clus, n_init = 22)
    clust_k_means.fit( data_df )

    #data_df.iloc[:,0:n_clus] = data_df.iloc[:,0:n_clus].astype(float)
    data_df = data_df.astype(float)
    data_df['Cluster'] = clust_k_means.labels_
    data_df_grp = data_df.groupby(['Cluster'], as_index = False ).mean()

    return data_df_grp, clust_k_means.labels_
```

1.7.3 Four Clusters

In [21]:
```
religion_sum, v_clust = Clustering_n(4, religion_df)
religion_sum.iloc[:,1:].plot(kind='bar', figsize=(6,16), subplots=True)
```

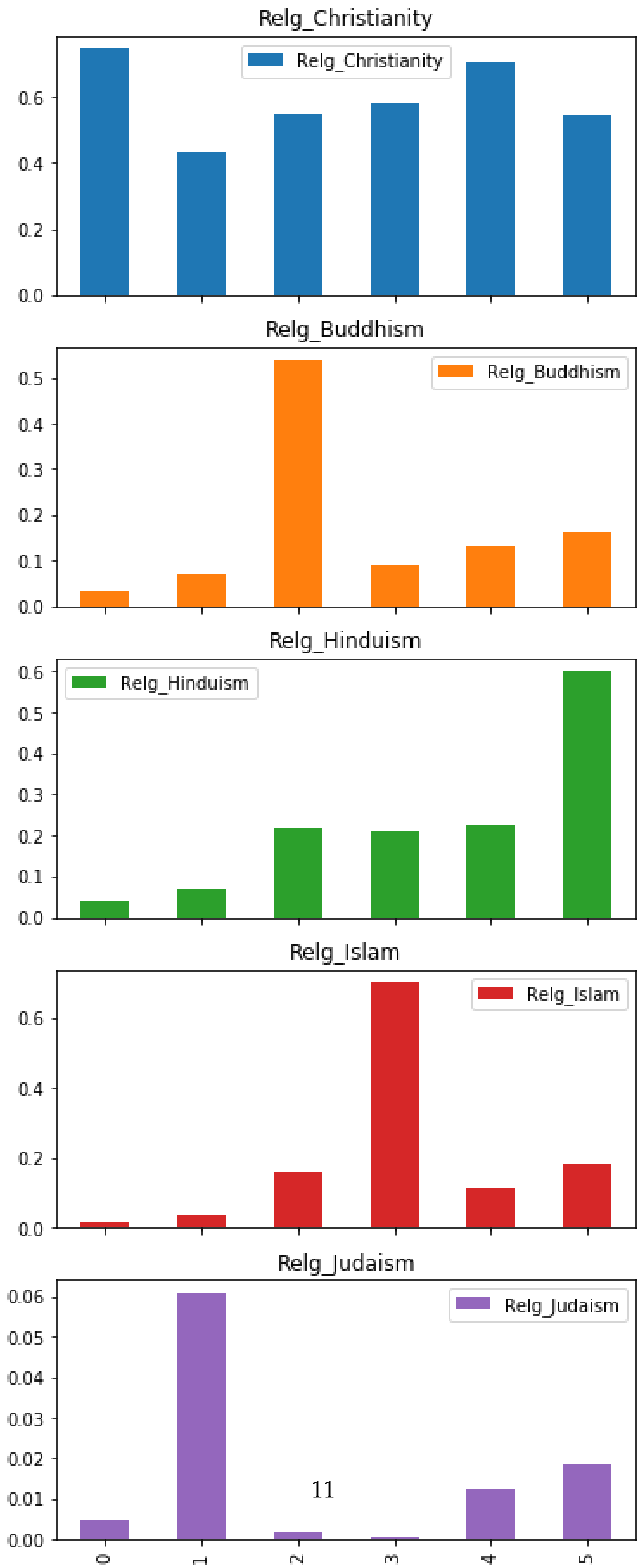
Out[21]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f9180837470>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f917fb24c88>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f918172bda0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f918096aeb8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f91820fdfd0>],
      dtype=object)
```


Relg_Christianity
Relg_Christianity
0.0
0.2
0.4
0.6
Relg_Buddhism
Relg_Buddhism
0.00
0.05
0.10
0.15
0.20
Relg_Hinduism
Relg_Hinduism
0.0
0.1
0.2
0.3
Relg_Islam
Relg_Islam
0.0
0.2
0.4
0.6
Relg_Judaism
Relg_Judaism
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
1
2
3

1.7.4 Six Clusters

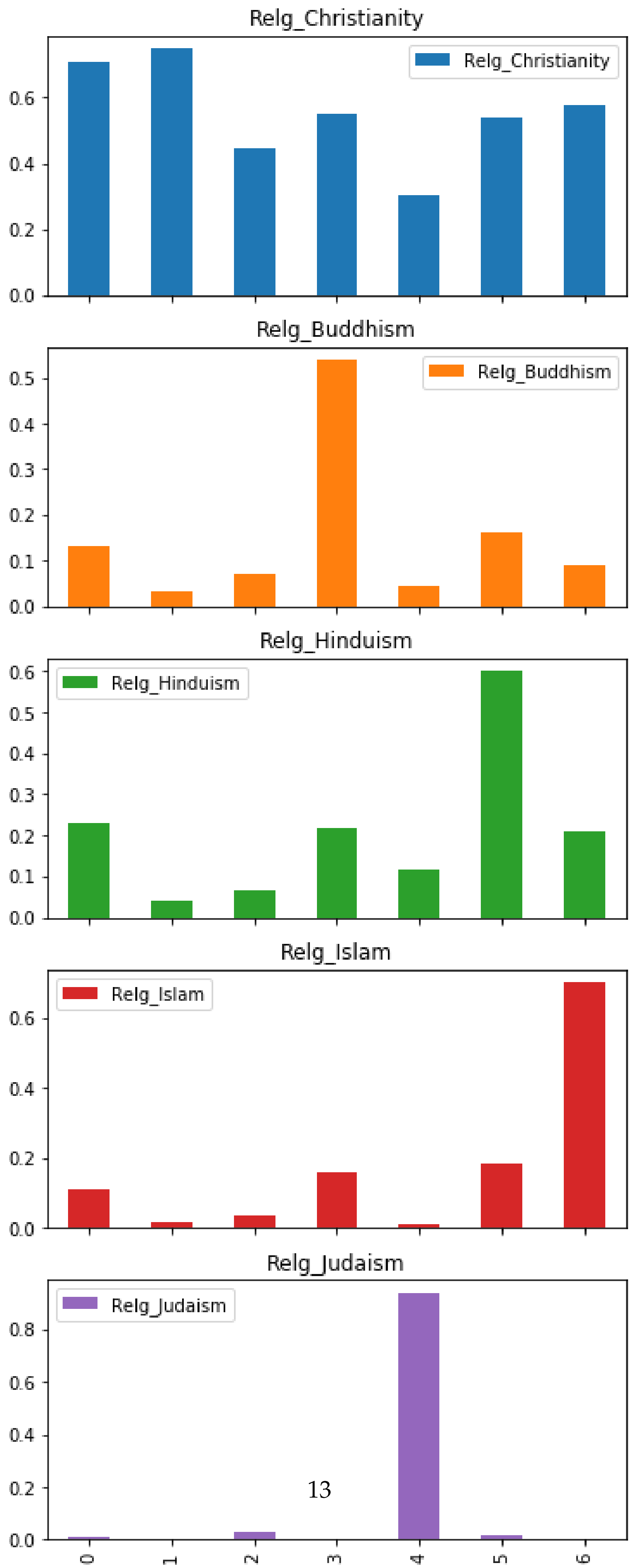
In [22]:
```
religion_sum, v_clust = Clustering_n(6, religion_df)
religion_sum.iloc[:,1:].plot(kind='bar', figsize=(6,16), subplots=True)
```

Out[22]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f91819da470>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f9181760f60>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f918132b400>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f9180e0f860>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f91816a1cc0>],
      dtype=object)
```


Relg_Christianity
Relg_Christianity
0.0
0.2
0.4
0.6
Relg_Buddhism
Relg_Buddhism
0.0
0.1
0.2
0.3
0.4
0.5
Relg_Hinduism
Relg_Hinduism
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Relg_Islam
Relg_Islam
0.0
0.2
0.4
0.6
Relg_Judaism
Relg_Judaism
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
1
2
3
4
5

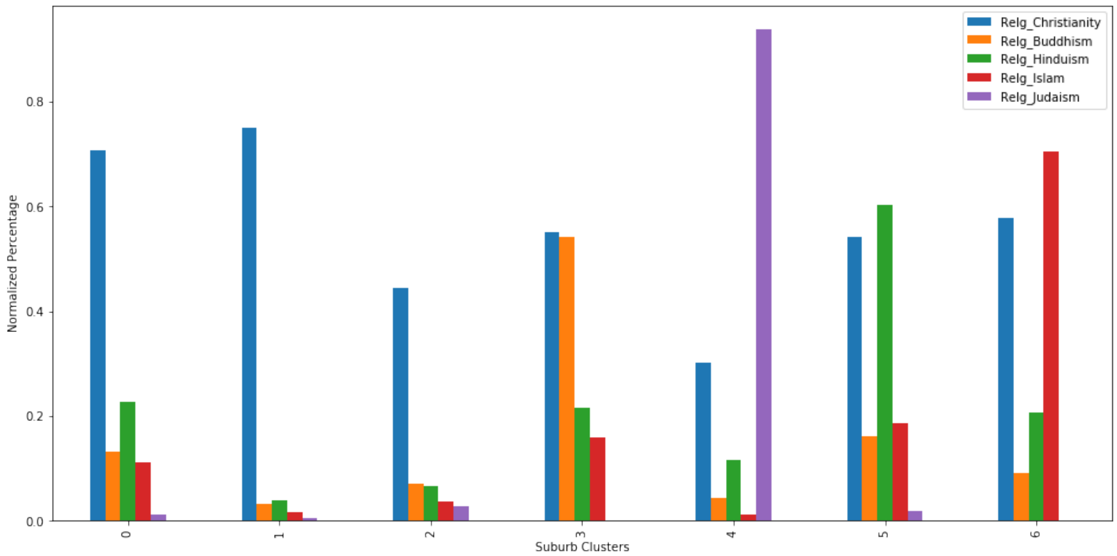
1.7.5 Seven Clusters

```
In [23]: religion_sum, v_clust = Clustering_n(7, religion_df)
         ax = religion_sum.iloc[:,1:].plot(kind='bar', figsize=(6,16), subplots=True)
```


Relg_Christianity
Relg_Christianity
0.0
0.2
0.4
0.6
Relg_Buddhism
Relg_Buddhism
0.0
0.1
0.2
0.3
0.4
0.5
Relg_Hinduism
Relg_Hinduism
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Relg_Islam
Relg_Islam
0.0
0.2
0.4
0.6
Relg_Judaism
Relg_Judaism
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
5
6

In [24]:
```
ax = religion_sum.iloc[:,1:].plot(kind='bar', figsize=(16,8), subplots=False)
ax.set_xlabel('Suburb Clusters')
ax.set_ylabel('Normalized Percentage')
```

Out[24]: Text(0, 0.5, 'Normalized Percentage')



In [25]:
```
rainbow = ['#00CC66','#FFFF00','#996699','#6699FF','#778855','#CC3300','#55FF33']

map_relg = folium.Map( location=[cbd_loc.latitude, cbd_loc.longitude],
                       zoom_start=11, min_zoom = 9, max_zoom=12 )

for i in range( religion_df.shape[0] ):
    suburb  = data.loc[i,['Suburb']][0]

    if suburb not in GeoLoc.index:
        continue

    cluster = v_clust[i]
    cluster_txt = str(cluster)
    colour  = rainbow[cluster]

    vPopUp  = cluster_txt + ':' + suburb
    geo_loc = [GeoLoc.loc[suburb][0], GeoLoc.loc[suburb][1]]

    folium.CircleMarker(geo_loc, 8, color=colour, fill=True, fill_color=colour,popup = vPopUp, fill_opacity

map_relg
```

Out[25]: <folium.folium.Map at 0x7f91809a7198>

1.8 Clustering based on The Country of Origin

In [26]:
```
cols = GetColumns('Born')
cols.remove('Born_Overseas')
cols
```

Out[26]:
```
['Born_in_Oceania_Ex_OZ',
 'Born_in_North_West_Europe',
 'Born_in_Southern_Eastern_Europe',
 'Born_in_North_Africa_Middle_East',
 'Born_in_South_East_Asia',
 'Born_in_North_East_Asia',
 'Born_in_Southern_Central_Asia',
 'Born_in_America',
 'Born_in_Sub_Saharan_Africa']
```

In [27]:
```
born_df = pd.DataFrame()

for i in range( len(cols) ):
    col_nm = cols[i]
    born_df[ col_nm ] = scale( data[ col_nm ].astype(float) )

    v_min   = born_df[ col_nm ].min()
    v_range = born_df[ col_nm ].max() - v_min
    born_df[ col_nm ] = ( born_df[ col_nm ] - v_min ) / v_range
```

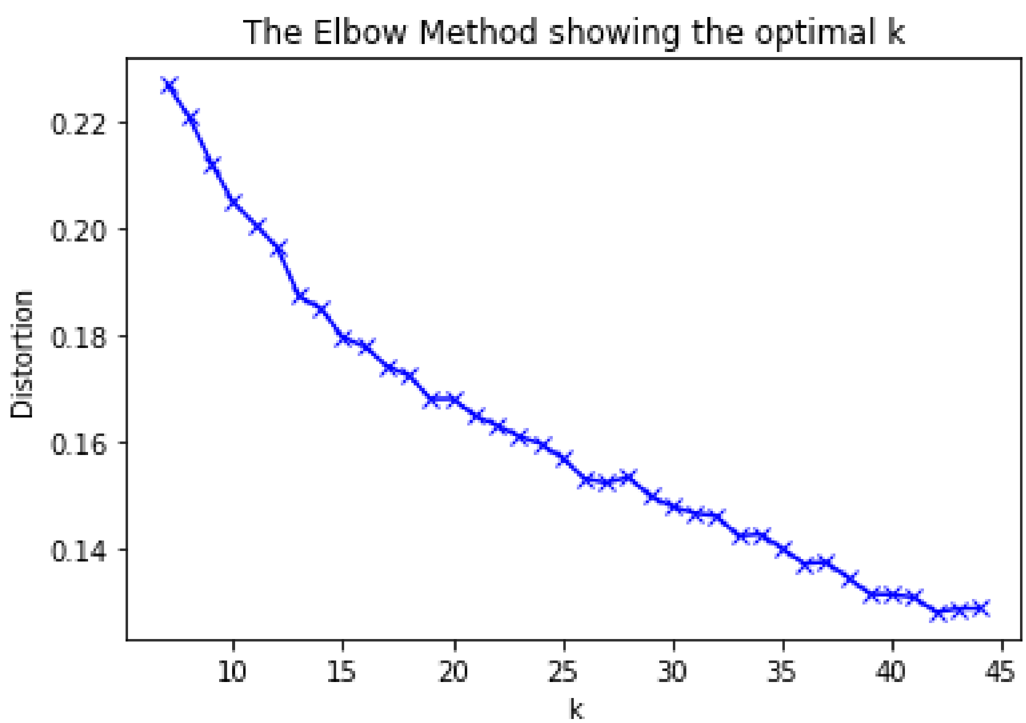
In [28]:
```
#data.drop( 'Born_Overseas', axis = 1)
born_df.head(3)
```

Out[28]:
```
   Born_in_Oceania_Ex_OZ  Born_in_North_West_Europe  \
0               0.164179                   0.215278
1               0.179104                   0.277778
2               0.104478                   0.250000

   Born_in_Southern_Eastern_Europe  Born_in_North_Africa_Middle_East  \
0                         0.039409                          0.013986
1                         0.039409                          0.010490
2                         0.039409                          0.006993

   Born_in_South_East_Asia  Born_in_North_East_Asia  \
0                 0.037037                 0.040230
1                 0.026455                 0.031609
2                 0.021164                 0.011494

   Born_in_Southern_Central_Asia  Born_in_America  Born_in_Sub_Saharan_Africa
0                       0.087774         0.076923                    0.145161
1                       0.050157         0.134615                    0.080645
2                       0.025078         0.115385                    0.064516
```

1.9 Find the Optimal k of Eblow Method

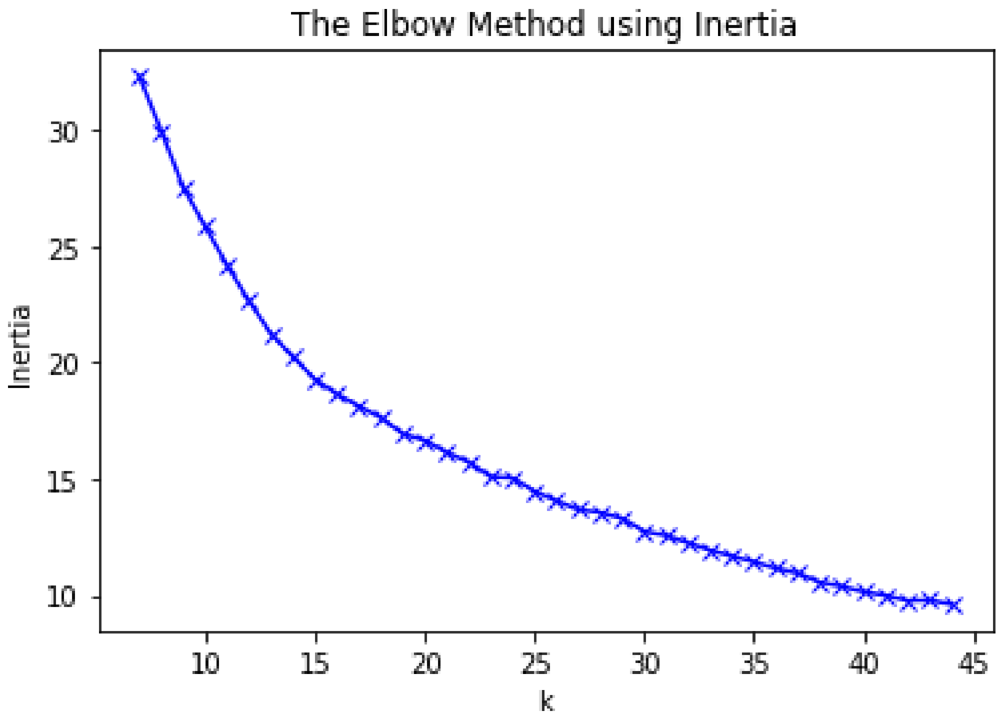
1.9.1 Using Distortion and Inertia

In [29]:
```
from scipy.spatial.distance import cdist

distortions = []
inertias = []
K = range(7,45)

for i in K:
    clust_k_means = KMeans(init = "k-means++", n_clusters = i, n_init = 22)
    clust_k_means.fit( born_df )
    distortions.append(sum(np.min(cdist(born_df, clust_k_means.cluster_centers_, 'euclidean'), axis=1))
    inertias.append(clust_k_means.inertia_)

# Plot the elbow
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



In [30]:
```
# Plot the elbow
```

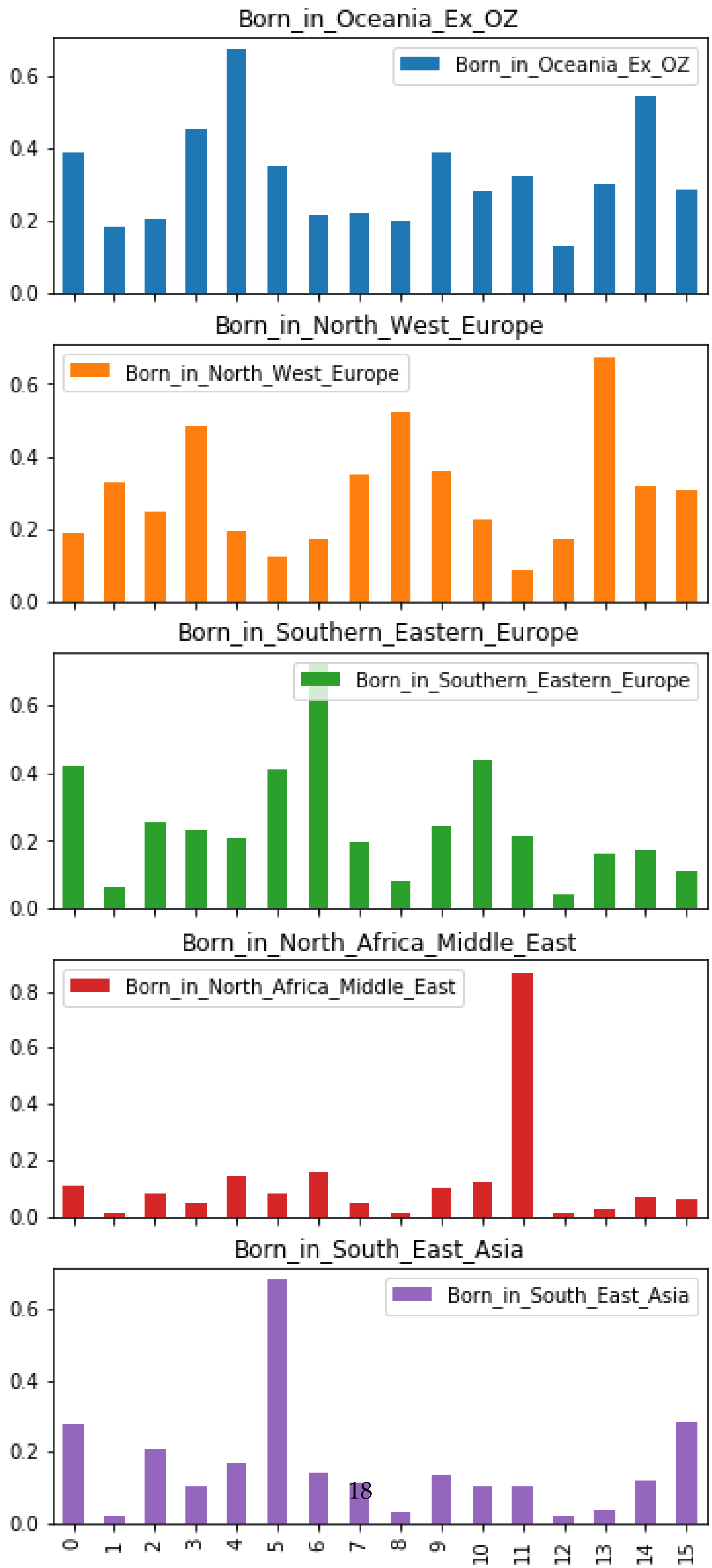
```
plt.plot(K, inertias, 'bx-')
plt.xlabel('k')
plt.ylabel('Inertia')
plt.title('The Elbow Method using Inertia')
plt.show()
```



1.9.2 Four Clusters Country of Origins - k = 16

In [31]:
```
cluster_k = 16
born_sum, v_clust = Clustering_n(cluster_k, born_df)
born_sum.iloc[:,1:6].plot(kind='bar', figsize=(6,14), subplots=True)
```

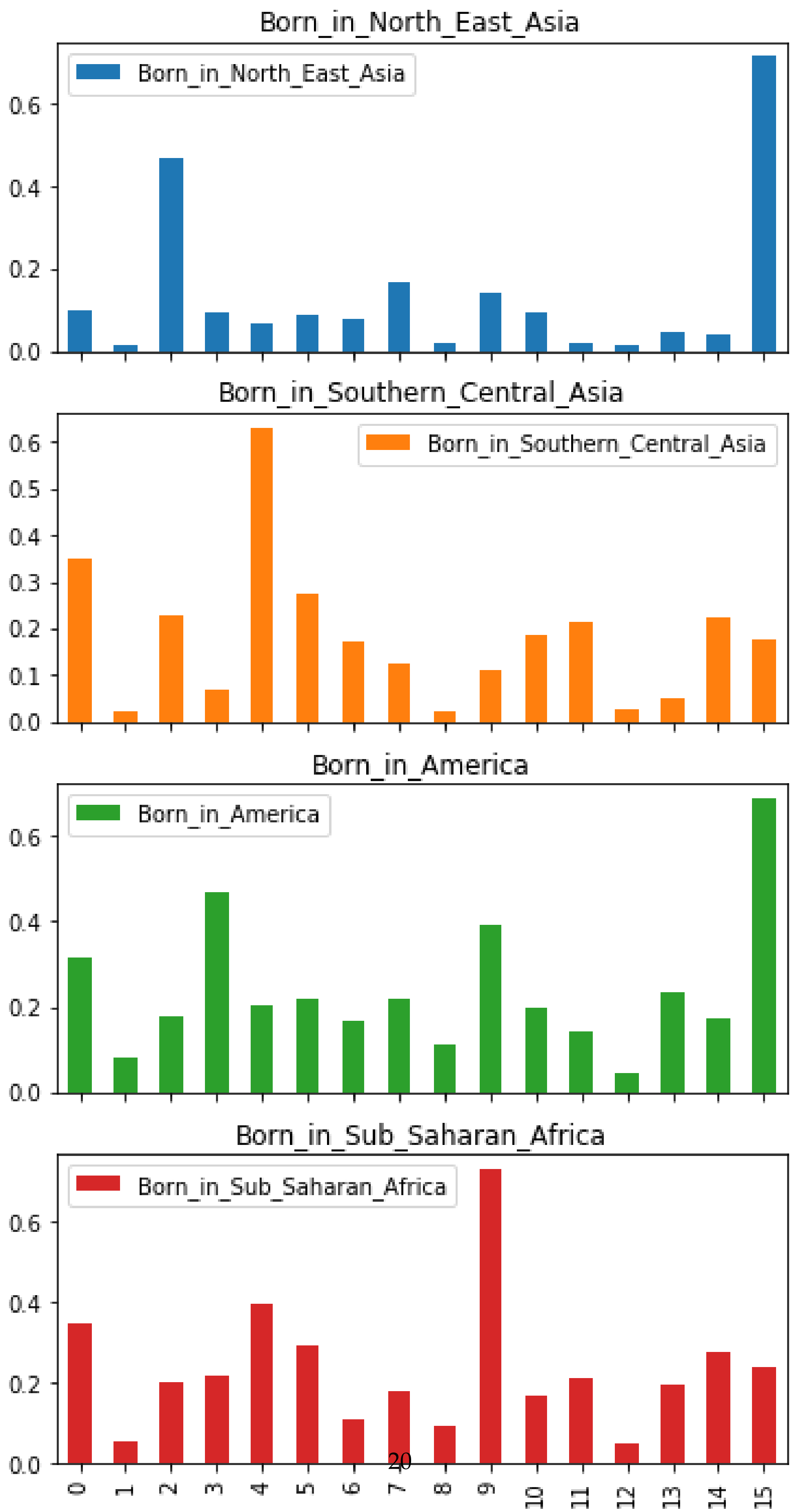
Out[31]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f91814cf898>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f9181c06438>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f917e337780>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f917decbbe0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f917feeb080>],
      dtype=object)
```



Born_in_Oceania_Ex_OZ
Born_in_North_West_Europe
Born_in_Southern_Eastern_Europe
Born_in_North_Africa_Middle_East
Born_in_South_East_Asia
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

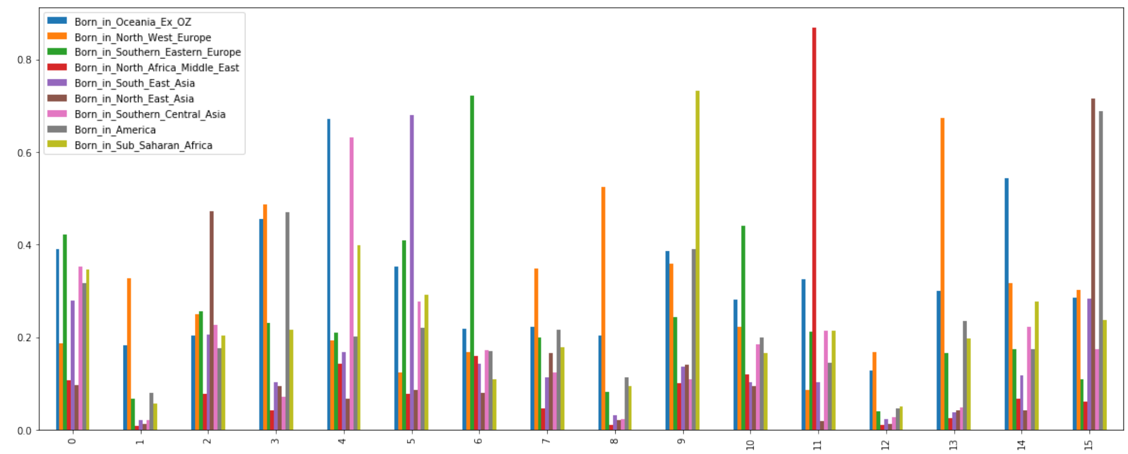
```
In [32]: born_sum.iloc[:,6:].plot(kind='bar', figsize=(6,12), subplots=True)
```

```
Out[32]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f917de42eb8>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f917e898940>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f917e0c8400>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7f9180161518>],
               dtype=object)
```


Born_in_North_East_Asia
Born_in_North_East_Asia
0.0
0.2
0.4
0.6
Born_in_Southern_Central_Asia
Born_in_Southern_Central_Asia
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Born_in_America
Born_in_America
0.0
0.2
0.4
0.6
Born_in_Sub_Saharan_Africa
Born_in_Sub_Saharan_Africa
0.0
0.2
0.4
0.6
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

In [33]: born_sum.iloc[:,1:].plot(kind='bar', figsize=(20,8), subplots=False)

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7f917e256048>



In [34]: born_sum

Out[34]:

	Cluster	Born_in_Oceania_Ex_OZ	Born_in_North_West_Europe \
0	0	0.389925	0.187066
1	1	0.182937	0.327140
2	2	0.203358	0.248843
3	3	0.455224	0.486111
4	4	0.672388	0.192708
5	5	0.353234	0.123264
6	6	0.217910	0.168519
7	7	0.222258	0.349034
8	8	0.202822	0.523592
9	9	0.385928	0.358135
10	10	0.282183	0.222873
11	11	0.324627	0.085069
12	12	0.129077	0.168853
13	13	0.300719	0.674383
14	14	0.542999	0.317460
15	15	0.286070	0.303241

	Born_in_Southern_Eastern_Europe	Born_in_North_Africa_Middle_East \
0	0.422722	0.107955
1	0.066769	0.009119
2	0.255337	0.077214
3	0.231351	0.043082

4	0.209606	0.142133
5	0.409278	0.078380
6	0.721511	0.160373
7	0.199293	0.046063
8	0.082597	0.010585
9	0.243490	0.100899
10	0.439655	0.119100
11	0.211823	0.868881
12	0.040777	0.009907
13	0.164933	0.026159
14	0.173352	0.067766
15	0.110016	0.061772

	Born_in_South_East_Asia	Born_in_North_East_Asia \
0	0.279762	0.096085
1	0.021879	0.011727
2	0.204696	0.471504
3	0.102891	0.095443
4	0.167196	0.067241
5	0.679894	0.085728
6	0.142504	0.079119
7	0.112664	0.165605
8	0.031057	0.021020
9	0.136810	0.140805
10	0.102100	0.095097
11	0.103175	0.018678
12	0.023908	0.012346
13	0.038213	0.042784
14	0.117536	0.041598
15	0.283951	0.715517

	Born_in_Southern_Central_Asia	Born_in_America	Born_in_Sub_Saharan_Africa
0	0.351685	0.317308	0.345766
1	0.020419	0.080042	0.056452
2	0.226489	0.176282	0.202957
3	0.070421	0.469780	0.216014
4	0.631818	0.201923	0.398387
5	0.276385	0.219551	0.291667
6	0.171996	0.169231	0.109677
7	0.124029	0.216973	0.178822
8	0.022888	0.114067	0.094344
9	0.110166	0.390110	0.732719
10	0.184757	0.199519	0.165827
11	0.214734	0.144231	0.213710
12	0.027575	0.045228	0.051075
13	0.048299	0.235043	0.197133
14	0.222571	0.173993	0.277266
15	0.175026	0.689103	0.236559

In [35]:
```python
# set color scheme for the clusters
x = np.arange(cluster_k)
ys = [i + x + (i*x)**2 for i in range(cluster_k)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]
```

In [36]:
```python
born_df.columns
```

Out[36]:
```
Index(['Born_in_Oceania_Ex_OZ', 'Born_in_North_West_Europe',
       'Born_in_Southern_Eastern_Europe', 'Born_in_North_Africa_Middle_East',
       'Born_in_South_East_Asia', 'Born_in_North_East_Asia',
       'Born_in_Southern_Central_Asia', 'Born_in_America',
       'Born_in_Sub_Saharan_Africa'],
      dtype='object')
```

In [37]:
```python
map_born = folium.Map( location=[cbd_loc.latitude, cbd_loc.longitude],
                       zoom_start=11, min_zoom = 9, max_zoom=12 )


for i in range( born_df.shape[0] ):
  suburb  = data.loc[i,['Suburb']][0]

  if suburb not in GeoLoc.index:
    continue

  cluster = v_clust[i]
  cluster_txt = str(cluster)
  colour  = rainbow[cluster]

  vPopUp  = cluster_txt + ':' + suburb
  geo_loc = [GeoLoc.loc[suburb][0], GeoLoc.loc[suburb][1]]

  folium.CircleMarker(geo_loc, 8, color=colour, fill=True, fill_color=colour,popup = vPopUp, fill_opacity


map_born
```

Out[37]:
```
<folium.folium.Map at 0x7f917e37be48>
```

In [ ]: