Cellphone Service Best Markets ¶

A. Importing

- Importing necessary libraries

In [199]:

```
import sqlite3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import folium
%matplotlib inline
```

- Import data

In [3]:

```
data = pd.read_csv('./wdi/Indicators.csv')
data.shape
```

Out[3]:

```
(5656458, 6)
```

B. Data preparation

In [132]:

```
data.columns
```

Out[132]:

```
Index(['CountryName', 'CountryCode', 'IndicatorName', 'IndicatorCod
e', 'Year',
       'Value'],
      dtype='object')
```

- List some basic statistics

In [134]:

```
#check for null values in any cell
data.isnull().any().any()
```

Out[134]:

False

In [245]:

```
countries = data['CountryName'].unique().tolist()
countryCodes = data['CountryCode'].unique().tolist()
indicators = data['IndicatorName'].unique().tolist()
indicator_codes = data['IndicatorCode'].unique().tolist()
```

In [5]:

```
print ('Countries:',len(countries),'  Indicators:',len(indicators))
```

Countries: 247   Indicators: 1344

- Create a data frame for Mobile cellular subscriptions

In [265]:

```
df_cellfone = data.loc[(data['IndicatorName'] == 'Mobile cellular subscriptions'
) & (data['Year'] >= 1995)]
df_cellfone = df_cellfone.drop(['IndicatorName', 'IndicatorCode'], axis=1)
df_cellfone.columns = ['CountryName','CountryCode','Year','Cellphone']
df_cellfone.head(3)
```

Out[265]:

	CountryName	CountryCode	Year	Cellphone
2485343	Arab World	ARB	1995	501551.0
2485757	Caribbean small states	CSS	1995	66297.0
2486225	Central Europe and the Baltics	CEB	1995	552442.0

- Create an empty dataframe to collect any feature that correlate well to mobile cellular subscriptions

In [266]:

```
df_features = pd.DataFrame({'IndicatorCode':['a'],'Correlation':[0.0],'Length':[
0]})
df_features
```

Out[266]:

	Correlation	IndicatorCode	Length
0	0.0	a	0

- Loop through the entire dataset and calculate correlation score. Features with correlation score greater or equal to 0.89 will be selected

In [268]:

```
for f in indicator_codes:
    temp = data.loc[(data['IndicatorCode'] == f) & (data['Year'] >= 1995)]
    temp = temp.drop(['IndicatorName', 'IndicatorCode'], axis=1)
    temp.columns = ['CountryName','CountryCode','Year', f]
    temp = df_cellfone.merge(temp, on=['CountryCode','Year'], how='inner')
    related = temp['Cellphone'].corr(temp[f])
    #print("%s: %f" % (f,related))
    if (related >= 0.85):
        df_features.loc[-1] = [related,f,len(temp)]
        df_features.index = df_features.index + 1
        df_features = df_features.sort_index()
df_features.shape
```

Out[268]:

```
(58, 3)
```

- Expand the df_features and in this case, we just add one more column - the "Indicator Name"

In [269]:

```
df_indicators = pd.DataFrame([indicators,indicator_codes])
df_indicators = df_indicators.transpose()
df_indicators.columns = ['Indicator Names', 'IndicatorCode']
df_features = df_features.merge(df_indicators, on=['IndicatorCode'])
```

- Here the features will be displayed. Note that while some correlation scores are high, they have very low "Length" values which indicates lesser confidence when compared to entries with same Correlation scores but with higher Length.

In [271]:

```
df_features_filtered = df_features.sort_values(by=['Correlation'])
df_features_filtered.tail(10)
```

Out[271]:

	Correlation	IndicatorCode	Length	Indicator Names
41	0.921564	DT.DOD.PRVS.CD	2559	External debt stocks, long-term private sector...
33	0.922053	DT.NFL.DECT.CD	2559	Net flows on external debt, total (NFL, curren...
39	0.923879	DT.DOD.DSTC.CD	2559	External debt stocks, short-term (DOD, current...
28	0.924596	DT.DOD.PNGC.CD	2559	PNG, commercial banks and other creditors (DOD...
50	0.932751	BX.TRF.PWKR.CD.DT	3793	Personal remittances, received (current US$)
54	0.944459	NV.AGR.TOTL.CD	4128	Agriculture, value added (current US$)
3	0.948240	BX.TRF.PWKR.CD	1583	Personal transfers, receipts (BoP, current US$)
0	0.955018	DT.DOD.PVLX.CD	117	Present value of external debt (current US$)
51	0.973193	ER.H2O.FWTL.K3	545	Annual freshwater withdrawals, total (billion ...
56	1.000000	IT.CEL.SETS	4683	Mobile cellular subscriptions

C. DATA EXPLORATION

Top 10 features based on both correlation and length (unverified)

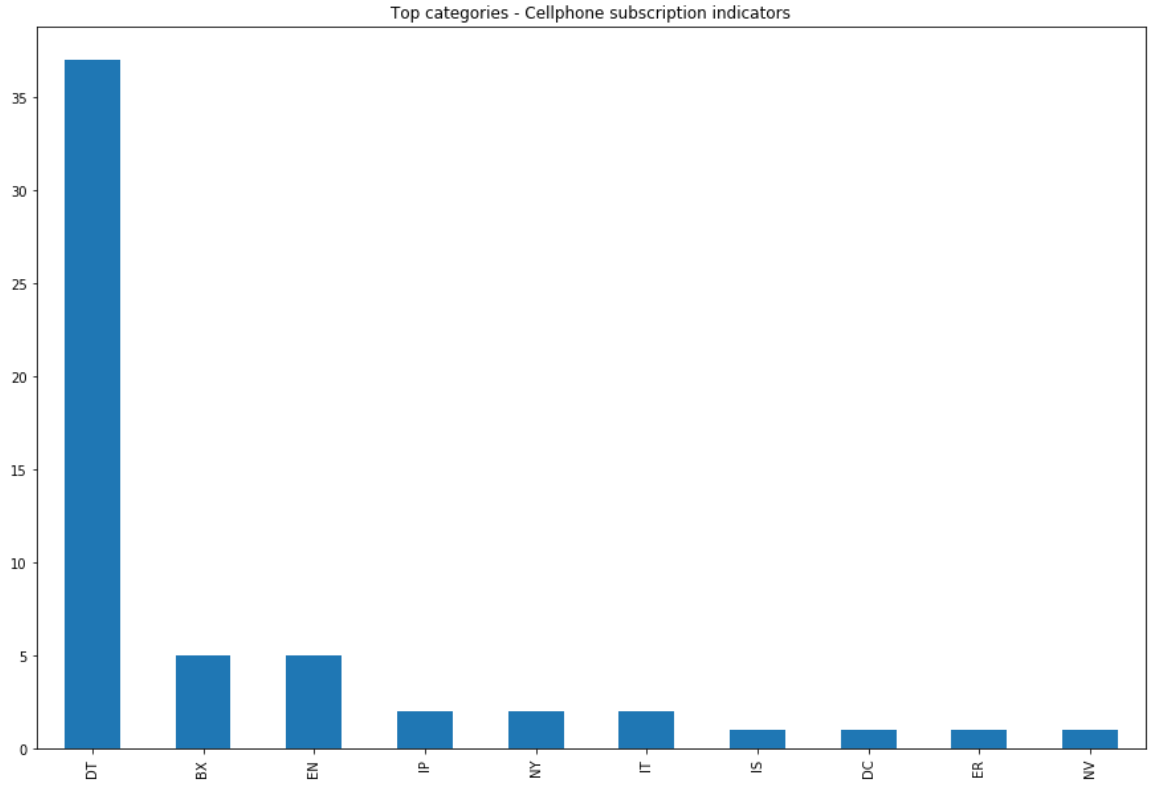
- Agriculture, value added : NV.AGR.TOTL.CD
- Personal remittances, received : BX.TRF.PWKR.CD.DT
- Net flows on external debt, total : DT.NFL.DECT.CD
- Container port traffic : IS.SHP.GOOD.TU
- Trademark applications, direct resident : IP.TMK.RESD
- Primary income on FDI, payments : BX.KLT.DREM.CD.DT
- HFC gas emissions (thousand metric tons of CO2) : EN.ATM.HFCG.KT.CE
- Fixed broadband subscriptions : IT.NET.BBND
- Grants, excluding technical cooperation : BX.GRT.EXTA.CD.WD
- GDP, PPP : NY.GDP.MKTP.PP.CD

In [272]:

```
df_features_codeCount = df_features['IndicatorCode'].str.split('.', expand=True)
df_features_codeCount.drop(df_features_codeCount.index[1:6], axis = 1, inplace=True)
df_features_codeCount = df_features_codeCount[df_features_codeCount.index[0]].value_counts()
df_features_codeCount.plot(kind='bar', figsize=(15,10), title='Top categories - Cellphone subscription indicators')
# https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html
```

Out[272]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a166092b0>
```



In [277]:

```
featured_codes=['NV.AGR.TOTL.CD','BX.TRF.PWKR.CD.DT','DT.NFL.DECT.CD','IS.SHP.GOOD.TU','IP.TMK.RESD','BX.KLT.DREM.CD.DT','EN.ATM.HFCG.KT.CE','IT.NET.BBND','BX.GRT.EXTA.CD.WD','NY.GDP.MKTP.PP.CD']
df_features_combined = df_cellfone
#df_features_combined.head(5)
```

In [278]:

```
for f in featured_codes:
    temp = data.loc[(data['IndicatorCode'] == f) & (data['Year'] >= 1995)]
    temp = temp.drop(['IndicatorName', 'IndicatorCode'], axis=1)
    temp.columns = ['CountryName','CountryCode','Year', f]
    df_features_combined = df_features_combined.merge(temp, on=['CountryName','C
ountryCode','Year'], how='inner')
    #get the relationship of each indicator with "Cellphone"
    df_features_combined[f] = df_features_combined[f]/df_features_combined['Cell
phone']
#drop "Cellphone" out of the df
df_features_combined = df_features_combined.drop('Cellphone', axis=1)
df_features_combined.head(10)
```

Out[278]:

	CountryName	CountryCode	Year	NV.AGR.TOTL.CD	BX.TRF.PWKR.CD.DT	DT.NFL
0	Brazil	BRA	2000	1343.901282	71.129154	192.81
1	China	CHN	2000	2084.989855	56.554546	-28.814
2	Colombia	COL	2000	3676.132271	713.453656	-102.35
3	Mexico	MEX	2000	1634.192265	534.494041	-1105.7
4	Peru	PER	2000	3248.144467	563.378635	222.447
5	Low & middle income	LMY	2005	823.229460	153.238552	87.082
6	Brazil	BRA	2005	480.634619	32.540854	-272.09
7	China	CHN	2005	676.352929	8.482488	101.139
8	Colombia	COL	2005	518.219107	153.117226	-10.793
9	Costa Rica	CRI	2005	1473.507061	381.680288	764.226

In [398]:

```
df_features_countryAvg = df_features_combined.groupby(['CountryName','CountryCod
e'], as_index=False).mean()
df_features_countryAvg = df_features_countryAvg.drop(['Year'], axis=1)

df_features_countryAvg.head(5)
```

Out[398]:

	CountryName	CountryCode	NV.AGR.TOTL.CD	BX.TRF.PWKR.CD.DT	DT.NFL.DECT
0	Albania	ALB	980.851852	616.656147	522.210978
1	Algeria	DZA	416.365198	5.997203	-6.979129
2	Bangladesh	BGD	324.861301	180.011626	29.378517
3	Brazil	BRA	700.083491	35.458936	124.693497
4	Cambodia	KHM	633.718833	31.533360	65.452955

In [399]:

```
df_features_countryAvg_scaled = df_features_countryAvg
for f in featured_codes:
    df_features_countryAvg_scaled[[f]] = MinMaxScaler().fit_transform(df_feature
s_countryAvg_scaled[[f]])
df_features_countryAvg_scaled.head()
```

Out[399]:

	CountryName	CountryCode	NV.AGR.TOTL.CD	BX.TRF.PWKR.CD.DT	DT.NFL.DECT
0	Albania	ALB	0.758783	0.796579	0.962113
1	Algeria	DZA	0.261615	0.000000	0.124866
2	Bangladesh	BGD	0.181024	0.226994	0.182388
3	Brazil	BRA	0.511498	0.038432	0.333189
4	Cambodia	KHM	0.453048	0.033311	0.239463

In [400]:

```
df_features_countryAvg_scaled_avg = df_features_countryAvg_scaled
df_features_countryAvg_scaled_avg['AVG'] = df_features_countryAvg_scaled_avg.mea
n(axis=1)
for f in featured_codes:
    df_features_countryAvg_scaled_avg.drop(f, axis = 1, inplace=True)
df_features_countryAvg_scaled_avg = df_features_countryAvg_scaled_avg.sort_value
s(by=['AVG'])
```

In [402]:

```
df_features_countryAvg_scaled_avg.tail(10)
```

Out[402]:

	CountryName	CountryCode	AVG
15	Jamaica	JAM	0.361667
31	Turkey	TUR	0.364619
18	Low & middle income	LMY	0.375248
6	Colombia	COL	0.387327
19	Malaysia	MYS	0.396094
20	Mexico	MEX	0.405907
0	Albania	ALB	0.414586
5	China	CHN	0.458202
25	Peru	PER	0.521901
7	Costa Rica	CRI	0.643711

In [346]:

```
country_geo = 'wdi/world-countries.json'
hist_indicator = df_features_countryAvg_scaled_avg.iloc[0]['AVG']
```

In [353]:

```
# Setup a folium map at a high-level zoom @Alok - what is the 100,0, doesn't see
m like lat long
map = folium.Map(location=[100, 0], zoom_start=1)
```

In [381]:

```
# choropleth maps bind Pandas Data Frames and json geometries.  This allows us t
o quickly visualize data combinations
map.choropleth(geo_path=country_geo, data=df_features_countryAvg_scaled_avg,
               columns=['CountryCode', 'AVG'],
               key_on='feature.id',
               fill_color='YlGn', fill_opacity=0.7, line_opacity=0.2,
               legend_name='Test'
              )
```

```
/Users/genterist/anaconda3/lib/python3.6/site-packages/ipykernel_lau
ncher.py:6: FutureWarning: 'threshold_scale' default behavior has ch
anged. Now you get a linear scale between the 'min' and the 'max' of
your data. To get former behavior, use folium.utilities.split_six.
```

In [385]:

```
map.save('plot.html')
#display(map)
#map.choropleth
```

```
---------------------------------------------------------------------
-------
KeyError                                  Traceback (most recent cal
l last)
<ipython-input-385-af76673efd86> in <module>()
----> 1 map.save('plot.html')
      2 #display(map)
      3 #map.choropleth

~/anaconda3/lib/python3.6/site-packages/folium/element.py in save(se
lf, outfile, close_file, **kwargs)
    151
    152         root = self.get_root()
--> 153         html = root.render(**kwargs)
    154         fid.write(html.encode('utf8'))
    155         if close_file:

~/anaconda3/lib/python3.6/site-packages/folium/element.py in render
(self, **kwargs)
    357         """Renders the HTML representation of the elemen
t."""
    358         for name, child in self._children.items():
--> 359             child.render(**kwargs)
    360         return self._template.render(this=self, kwargs=kwarg
s)
    361

~/anaconda3/lib/python3.6/site-packages/folium/element.py in render
(self, **kwargs)
    665
    666         for name, element in self._children.items():
--> 667             element.render(**kwargs)

~/anaconda3/lib/python3.6/site-packages/folium/element.py in render
(self, **kwargs)
    661         script = self._template.module.__dict__.get('script'
, None)
    662         if script is not None:
--> 663             figure.script.add_children(Element(script(self,
 kwargs)),
    664                                        name=self.get_name())
    665

~/anaconda3/lib/python3.6/site-packages/jinja2/runtime.py in __call_
_(self, *args, **kwargs)
    547                                 (self.name, len(self.argument
s)))
    548
--> 549         return self._invoke(arguments, autoescape)
    550
    551     def _invoke(self, arguments, autoescape):

~/anaconda3/lib/python3.6/site-packages/jinja2/asyncsupport.py in _i
```