my_surroundings

March 8, 2018

1 My Surroundings

1.1 Author: Gregorio Ferreira - ferreiradesajg@gmail.com

1.1.1 Document description:

This notebook, contains the steps followed to explore, validate and prepare the "Surroundings.json" data source, to be used as features.

```
In [1]: import json
        import pandas as pd
        from pandas.io.json import json_normalize #package for flattening json in pandas df
        import numpy as np
        import os
        os.chdir('XXX - YOUR PATH HERE - XXX')

        #load json object
        with open('./data/Surroundings.json') as my_list:
            my_list = json.load(my_list)

        print('Sanity check 1, there are only two keys:store_code, surroundings.\nNo list shoul
        for x in my_list:
            if len(x) != 2:
                print(x)
```

```
Sanity check 1, there are only two keys:store_code, surroundings.
No list should be returned if all ok!
```

```
In [2]: ## Extracting surroundings details
        colNames = ('store_code','col_name','name', 'place_id', 'latitude', 'longitude', 'count

        # Define a dataframe with the required column names
        df_out = pd.DataFrame(columns = colNames)

        for x in my_list:
            my_surr = json_normalize(x['surroundings'])
```

```
        cols = list(my_surr.columns)
        #col = 84
        for col in cols:
            my_list_l2 = json_normalize(x['surroundings'][col])

            if(len(my_list_l2) != 0):

                    for ii in range(0, len(my_list_l2)):
                        temp = json_normalize(my_list_l2.address_components[ii])

                        postal_code_test = temp.loc[temp['types'].astype(str) == "['postal_

                        if(len(postal_code_test) != 0):
                            postal_code = postal_code_test.values[0]
                        else:
                            postal_code = np.nan

                        country_test = temp.loc[temp['types'].astype(str) == "['country',

                        if(len(country_test) != 0):
                            country = country_test.values[0]
                        else:
                            country = np.nan

                        df_out = df_out.append([{'store_code':x['store_code'],
                                    'col_name' : col,
                                    'name' : my_list_l2.name[ii],
                                    'place_id' : my_list_l2.place_id[ii],
                                    'latitude' : my_list_l2.latitude[ii],
                                    'longitude' : my_list_l2.longitude[ii],
                                    'postal_code' : postal_code,
                                    'country' : country,
                                    }],ignore_index=True)
```

In [ ]:
```
surroundings_details = df_out[['store_code', 'name', 'place_id', 'latitude', 'longitude
                               'country', 'postal_code']].drop_duplicates(keep='first')

surroundings_details.to_csv("./result dataset/surroundings_details.csv",
                            sep=',', encoding='utf-8',index=True)

surroundings_count = df_out[['store_code', 'col_name']].pivot_table(index = 'store_code
                           columns = 'col_name', aggfunc = len, fill_value = 0)

surroundings_count.to_csv("./result dataset/surroundings_count.csv",
                          sep=',', encoding='utf-8', index=True)

repeated_names = surroundings_details.groupby(['name'])['name'].agg(['count'])
```

```
repeated_names = repeated_names[(repeated_names['count'] > 1)]

repeated_names.to_csv("./result dataset/repeated_names.csv",
                      sep=',', encoding='utf-8', index=True)
```

In [ ]: