NB

February 17, 2018

1 Introduction

In this problem,We have to train a model to identify unique patients in the sample dataset given.We are given with a csv file which contains Name,gender,DOB and father's name.As data can come from various sources,there is great possiblity of data redundancy.So we have to train our model to remove that redundancy by checking out those datas which are similar and belongs to same person.So for solving this problem,I have used PYTHON and used packages like numpy,pandas and sklearn.

```
In [2]: import warnings
        warnings.filterwarnings("ignore")
        import pandas as pd
        import numpy as np
        import time
        import datetime
        from sklearn import preprocessing, cross_validation, svm, metrics
        from sklearn.cluster import KMeans

        df = pd.read_csv("data/train.csv")
        df.head()
```

Out[2]:

	ln	dob	gn	fn
0	SMITH JR	01/03/68	F	WILLIAM
1	ROTHMEYER JR	01/03/68	F	WILLIAM
2	ASBY JR	01/03/68	F	WILLIAM
3	SALTER JR	01/03/68	F	WILLIAM
4	SALTER JR	01/03/68	F	WILLIAM

2 Data Preprocessing:

So,as given data is in very raw form,for applying algorithm,we have to convert it into suitable form.So here we are reading the data from given csv file -train.csv which is present in data subfolderAs we have to convert data into suitable datatype,we are counting the number of letters 1in fathers name (fn) and then will be sorting the data in descending order of name.(fn).To make cluster of people with simillar fathers name,we will assign a lable to fn_len,and we will change them for people with different fathers name.To differentiate people with different first names,we will do same thing on (ln) and get (ln_lable). To convert gender into suitable datatype,we will be

assigning 0 to male and 1 to female. To convert DOB into suitbale datatype,we will be using unix date format that has unique integer assigned to every day in any year.

```
In [3]: df['fn_len'] = df['fn'].apply(lambda val:len(val))
```

```
In [4]: df.head()
```

```
Out[4]:              ln       dob gn       fn  fn_len
        0      SMITH JR  01/03/68  F  WILLIAM       7
        1  ROTHMEYER JR  01/03/68  F  WILLIAM       7
        2       ASBY JR  01/03/68  F  WILLIAM       7
        3     SALTER JR  01/03/68  F  WILLIAM       7
        4     SALTER JR  01/03/68  F  WILLIAM       7
```

```
In [5]: df.sort_values(['fn'],ascending=False,inplace=True)
        df.reset_index(inplace=True)
        df.drop(['index'],1,inplace=True)
```

```
In [6]: j = 0
        df['fn_label'] = df['fn_len']
        df['fn_label'][0] = 0
        for i in range(1,len(df)):
            if(df['fn'][i] in df['fn'][i-1]):
                df['fn_label'][i] = df['fn_label'][i-1]
            else:
                j += 1
                df['fn_label'][i] = j
```

```
In [7]: df.head()
```

```
Out[7]:             ln       dob gn       fn  fn_len  fn_label
        0    SMITH JR  01/03/68  F  WILLIAM       7         0
        1   BLAND III  21/02/62  F  WILLIAM       7         0
        2  SHAFFER JR  25/10/53  F  WILLIAM       7         0
        3    BLAND JR  25/10/53  F  WILLIAM       7         0
        4    BLAND JR  25/10/53  F  WILLIAM       7         0
```

```
In [8]: df['ln_len'] = df['ln'].apply(lambda val:len(val))
        df.sort_values(['ln'],ascending=False,inplace=True)
        df.reset_index(inplace=True)
        df.drop(['index'],1,inplace=True)

        j = 0
        df['ln_label'] = df['ln_len']
        df['ln_label'][0] = 0
        for i in range(1,len(df)):
            if(df['ln'][i] in df['ln'][i-1]):
                df['ln_label'][i] = df['ln_label'][i-1]
            else:
                j += 1
                df['ln_label'][i] = j
```

```
In [9]: df.head()
```

Out[9]:

	ln	dob	gn	fn	fn_len	fn_label	ln_len	ln_label
0	SMITH JR	01/03/68	F	WILLIAM	7	0	8	0
1	SMITH JR	07/10/37	M	HAROLD	6	13	8	0
2	SHAFFER JR	25/10/53	F	WILLIAM	7	0	10	1
3	SHAFFER JR	21/02/62	F	WILLIAM	7	0	10	1
4	SHAFFER JR	21/02/62	F	WILLIAM	7	0	10	1

```
In [10]: df["val_gn"] = df["gn"].map({"M":0,"F":1})

In [11]: def DOB_to_unix(str):
             s = str.split('/')
             str = s[0] + '/' + s[1]+ '/19' + s[2]

             return int((time.mktime(datetime.datetime.strptime(str, "%d/%m/%Y").timetuple())))

         df["val_dob"] = df["dob"].apply(lambda x: DOB_to_unix(x))

In [12]: df.head()
```

Out[12]:

	ln	dob	gn	fn	fn_len	fn_label	ln_len	ln_label	\
0	SMITH JR	01/03/68	F	WILLIAM	7	0	8	0	
1	SMITH JR	07/10/37	M	HAROLD	6	13	8	0	
2	SHAFFER JR	25/10/53	F	WILLIAM	7	0	10	1	
3	SHAFFER JR	21/02/62	F	WILLIAM	7	0	10	1	
4	SHAFFER JR	21/02/62	F	WILLIAM	7	0	10	1	

	val_gn	val_dob
0	1	-57994200
1	0	-1017293400
2	1	-510816600
3	1	-248074200
4	1	-248074200

```
In [13]: train = np.array(df[['fn_label','ln_label','val_gn','val_dob']])
         scaler = preprocessing.StandardScaler().fit(train)
         train = scaler.transform(train)
         train.shape
```

```
Out[13]: (103, 4)

In [14]: train[:5]

Out[14]: array([[-1.3825181 , -1.67527297,  1.7209121 ,  2.39653414],
                [ 0.60161023, -1.67527297, -0.5810872 , -0.82671534],
                [-1.3825181 , -1.50774567,  1.7209121 ,  0.87504891],
                [-1.3825181 , -1.50774567,  1.7209121 ,  1.75786452],
                [-1.3825181 , -1.50774567,  1.7209121 ,  1.75786452]])
```

3 Learning Algorithm:

Here in this problem,I will be using K means clustering algo.Initially we will be forming n number of clusters which are equal to number of different elements in training set -train.csv.We will give training points to fit() function and will make groups using predict() function.55 (printed o/p)is number of different final clusters remained after deduplication.After sorting,we will drop the rows having duplicate values and will only take one value per cluster in our final type.csv file.

```
In [15]: n_cluster = len(train)
         kmn = KMeans(n_clusters=n_cluster)
         kmn.fit(train)
         groups = kmn.predict(train)
         out = df
         out['type'] = groups
         print(len(out['type'].unique()))
```

```
55
```

```
In [16]: out.sort_values(['type'],ascending=False,inplace=True)
         out.drop_duplicates('type')
         out.reset_index(inplace=True)
         out.drop(['index'],1,inplace=True)
         out.head()
```

Out[16]:

	ln	dob	gn	fn	fn_len	fn_label	ln_len	ln_label	val_gn	\
0	MELVIN JR	07/10/37	M	HAROLD	6	13	9	6	0	
1	MELVIN JR	07/10/37	M	HAROLD	6	13	9	6	0	
2	MELVIN JR	07/10/37	M	HAROLD	6	13	9	6	0	
3	MELVIN JR	07/10/37	M	HAROLD	6	13	9	6	0	
4	MELVIN JR	07/10/37	M	HAROLD	6	13	9	6	0	

	val_dob	type
0	-1017293400	55
1	-1017293400	55
2	-1017293400	55
3	-1017293400	55
4	-1017293400	55

```
In [17]: out.reset_index(inplace=True)
         out.drop(['index'],1,inplace=True)
         out.head()
         out = df[['ln','dob','gn','fn']]
```

```
In [18]: out.to_csv('type.csv',index=None)
```