Finding Unique Patterns in Dialysis Facilities with Patients' Data using Unsupervised Learning Algorithms

Capstone project for Data Science Career Track bootcamp

Benhur Tedros

Summary

Kidney failure (ESRD: end stage renal disease) is one of the leading cause of death in the United States. According to USRDS 2013 Annual data report, this disease affects almost 650,000 people per year in US, and its rate is increasing by 5% each year. Today, ESRD patients have two treament options, which are either kidney transplantation or dialysis. The best current treament is the transplantation; however, the number of kidney donors to ESRD patients ratio is 1 to 6. Moreover, the need for kidney is increasing at 8% per year while their availability has not grown to match up that number. Therefore, dialysis is the only alternative option that the patients on waiting list have. People undergoing dialysis often have multiple health concerns,which can have an adverse impact on their life expectancy, though dialysis may offer a better quality of life. According to the National Institute of Diabetes and Digestive and Kidney Diseases report, the two-year, five year and ten-year survival rates are around 64%, 33%, and 10% respectively. The survival or mortality rate varies from one dialysis facility to another. In light of this, "could we find groups of dialysis facilities with similar behaviour given a dataset of the patients' health records that have been collected during their visits?" For example, would the effect be better if the facility use hemodialysis for patient A and peritoneal dialysis for patient B or vice versa.

There are several dialysis facilities registered with Medicare in the country where patients often visit. Besides the importance of the quality of care they provide, additional health data of their patients is collected. Some of the data include mortality rate (deaths), hospitalizations, blood transfusions, incidents of hypercalcemia (too much calcium in the blood), percentage of waste removed during hemodialysis in adults and children, percentage of waste removed in adults during peritoneal dialysis, percentage of AV fistulas, percentage of catheters in use over 90 days and others.

This capstone project will try to find a meaningful pattern within the health data of the patients and cluster the facilities with similar behaviour into groups. The result of this work will primarily help the dialysis facilities to improve their services. The Medicare department of the Federal government will also be benefited from the result of this project.

Objective of this project

The goal of this capstone project is:

- Organize the dialysis facilities into groups in which the facilities in each cluster are similar in some way

Data

The data for this project was publised by Centers for Medicare & Medicaid Services and was downloaded from the DATA.MEDICARE.GOV. The dataset is comprised of data on anemia management, phosphorus levels, transfusion rate, dialysis adequacy, vascular access, mineral and bone disorder, hospitalization rate, readmission rate,infection ratio, scale rate of the facility and others.The data was collected from 2012 to 2015 and can be downloaded from:

https://data.medicare.gov/Dialysis-Facility-Compare/Dialysis-Facility-Compare-Listing-by-Facility/23ew-n7w9/data

The details of the data fields with their term definitions can be found at the following link by clicking the " get supporting documents tab":

https://data.medicare.gov/data/dialysis-facility-compare

Two additional dataset were also used to merge with the original one. The first dataset includes population size for each counties, and the second one has an information on household income based on counties. The dataset were downloaded from the websites which are hyperlinked below.

https://www.census.gov/data/datasets/2016/demo/popest/counties-detail.html
https://en.wikipedia.org/wiki/List_of_United_States_counties_by_per_capita_income

Methodology

This project will be treated as unsupervised learning classification problem. Data loading, data wrangling and cleaning, feature selection, exploratory data analysis, inferential statistics, matrix manipulation, data visualization, clustering, model evaluation were carried out one after another to achieve the objectives of the project.The libraries used in this project include numpy, pandas, matplotlib, scikit-learn and Seaborn.

A. Data Wrangling and Cleaning

The dataset are stored in MS Excel spreadsheet in CSV format, which were easily loaded into pandas dataframes. Several components of data wrangling were utilized to make the dataset ready to be used for clustering. The dataset was originally in the format of CSV and loaded to the python notebook. It contains 6810 observations with 98 data fields with some missing/null values. Many of the names of the data fields were too long and were shortened for our convenience. The dataset was subset into fewer data fields/features depending on their scalability nature and importance to our prediction. Some of the columns in the dataset contained categorical variables, but were converted into numerical nature in order to get them included in the analysis. As this dataset was short of information on population size and household income in each county where the dialysis facilities are located, two additional dataset with the same data were merged to the original dataset before conducting the exploratory data analysis.

B. Feature Selection

The dataset has 6810 observations with 98 data fields with some null valuesdataset and contains some non-scalable and categorical variables/ non-numerical labels. The non-scalable data fields were dropped, and the others with the categorical variables were converted into numerical variables in order not to miss important information from those attributes. Population size and household income datafields are extracted from other datasets.

Forty six data fields/features were selected to be utilized in the model prediction.

C. Exploratory data Analysis (EDA) and Inferential Statistics

Explaratory data analysis for this dataset is useful in determining relationships among the explanatory variables, preliminary selection of appropriate models and assessing the direction and rough size of relationships among explanatory variables. This section will try to answer if there is a significance in terms of explaining unique pattern among the features given the health data of patients in the dialysis facilities across the States. Moreover, are there strong correlations between pairs of facilities or States? Before hand, it is vital to explore the distribution of the some of the features such as Mortality_rate, five_star (which is 1-5 scale reviews for the facilities) across the States. The mean and median of the Mortality_rate and five_star variables across the States will be explored. This will help us to explore the variations and similarities among the dialysis facilities with the same or different States.

Hypothesis:

Some of the health data such as readmission, hospitalization, standard infection rates are believed to show a strong positive correlations to the Mortality_rate. Moreover, patients with arteriovenous_fistulae has a lower risk of infection than patients with catheters, and so does their Mortality_rate. Therefore, exploring mortality rate across each States is vital,which is believed to show some trend. The five_star variable is also expected to display some variations.

Null Hypothesis,

- Ho: There is no correlation between the pairs of dialysis facilities across the States given the health data of patients in the dialysis facilities across the States.

Alternative Hypothesis,

- Ha:There is correlation between the pairs of the dialysis facilities

This hypothesis will be tested in this section.

D. Model Fitting

Clustering algorithms were utilized for this project. Each algorithm implements fit method to learn the clusters on train data and returns an array of integer labels corresponding to the different clusters.

K-Means

This algorithm is simple and cluster data by trying to separate samples in n groups of equal variance (minimizing within-cluster sum-of-squares). Number of clusters has to be specified for this algorithm. A range of number of clusters (2 to 10) were used and their results were evaluated using the silhouette coefficient. From the EDA of the dataset, it can be obserbed that the dataset can be categorized into four groups. Therefore, number of clusters of 4 was selected, though the silhouette score were less than that of cluster-2 and 3.

Agglomerative clustering

Agglomerative clustering algorithms build nested clusters by merging or splitting them successively. This algorithm also takes an input for number of clusters. The same range in number of clusters were utilized as of the K-Means. The result of using each number in the algorithm was evaluated using silhouette coefficient.

Affinity propagation

Affinity propagation builds clusters by sending messages between pairs of samples until convergence. The interesting part of this algorithm is that it chooses the number of clusters based on the data provided. It does not need to specifiy the number of clusters.

DBSCAN

The DBSCAN is a density based algorithm and sees clusters as areas of high density separated by areas of low density. It extracts the dense clusters and leaves sparse background classified as 'noise'.

Model Evaluation

The dataset was split into 50% of training and 50% testing dataset. The goal to split into two dataset is to observe if their result can be similar and evaluate the model performance. The above four clustering algorithms are fit on the training data, and the performance of each model was evaluated by the silhouette score.

The model with better performance was selected to be utilized on the left out 50% of the dataset (testing data).

Results

Selected clustering models were utilized to achieve the objective of this project. These models were fit on the train dataset (50% of the whole dataset). Some of them such as K-Means and Agglomerative clustering require number of clusters to be specified. However, the Affinity Propagation and DBSCAN do not have an input for number of clusters. The silhouette coefficient for each model seemed to have different values. While K-Means and Agglomerative clustering were indicating a reasonable to strong structure/pattern in the train dataset, the Affinity Propagation and

DBSCAN models were not effective in finding a substantial structure in the train dataset. As the result of K-Means and Agglomerative clustering was similar, K-Means model were selected for further analysis because of its simplicity.

The Silhouette score for different number of clusters used in the K-Means model shows values ranging from 0.85 to 0.61. These values can indicate that there is a reasonable to strong structure/pattern in the dataset. Though the number of cluster of 2 yielded the highest silhouette score (0.88), the number of cluster of 4 with a value of 0.70 was selected. The choice for that number was dependent on the exploratory data analysis carried out between the mortality rate/five star reviews and the States. On both cases, the States appeared to be categorized into four groups based on those data fields.

As part of model evaluation, the K-Means was fit on the 50 % left out dataset (testing dataset). Similar to the training dataset, the number of cluster of 4 with the silhouette score of 0.72 were estimated. The observations in each cluster were further analyzed based on their State level. From the clustering analysis, 5248, 1007, 384 and 171 observations (dialysis facilities) were grouped into cluster-0, cluster-3, cluster-2 and cluster-1 respectively.

Cluster - 0:

About 77 % of the dialysis facilities are clustered in this category. Each State contributed some of its facilities in this cluster.

Cluster - 1:

Only facilities from California were included into cluster-1, and it holds 3% of the dialysis facilties.

Cluster - 2:

About 6% of the facilities are part of this cluster. Cluster-2 only holds some facilities from California, Illinois, Texas and Arizona.

Cluster - 3:

This is the second cluster containing about 14% of the observations. In cluster-3, some dialysis facilities from California, Texas, Florida, Ohio, Pennsylvania, Minnesota, New York, Michigan, Washington, Utah, Virginia, Hawaii, and Montana are present.

Limitations

The result of this project does not answer why the dialysis facilities from different States clustered into one or the other. For example, the dialysis facilities from California are clustered into different groups, though they are from the same state. What made them differn or similiar from each other? The clustering models do not answer these questions which are very important. The models simply examine for any significant structure/pattern in the dialysis facilities dataset which helps to learn more about the data and for further analysis.

Further Research

The results are a preliminary product to our virtual client. Further research is needed to hand over a final product which will have a concrete and labeled variables. The dataset in each cluster should be analysed separately using supervised learning alogrithms.

Client Recommendations

As the model identified some structure in the dataset, some recommendations can be taken from the carried out analysis.

- Our client should investigate more the similarities and differences among the facilities clustered in one to the others respectively.

- The dialysis facilities in one State, but categorized into different clusters, should learn from each other.

- As this project is a preliminary analysis report, further research is needed based on the result of this project. The baseline for the next research should be the preliminary results of this project.
- The dialysis facilities should continue gathering more health data of the patients.

Code

In [1]:
```
# Importing the required libraries

%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
import os
```

In [2]:
```
# Setting up the path directory
os.chdir('F:\\BENHUR FOLDER\\Data Science Career Track\\Capstone Project')
os.getcwd()
```

Out[2]:
```
'F:\\BENHUR FOLDER\\Data Science Career Track\\Capstone Project'
```

In [3]:
```
# Loading the csv datafile into pandas dataframes
data = pd.read_csv('Dialysis_Mortality.csv')

# Getting information on the data fields, attributes,data types,field names
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6810 entries, 0 to 6809
Data columns (total 98 columns):
Provider Number
                6810 non-null object
Network
                6810 non-null int64
Facility Name
                6810 non-null object
Five Star Date
                6810 non-null object
Five Star
                6017 non-null float64
Five Star Data Availability Code
                6810 non-null int64
Address Line 1
                6810 non-null object
Address Line 2
                1014 non-null object
City
```

In [4]:
```
# Subsetting the features needed for the prediction
feature_select= pd.DataFrame(data.iloc[:,[1,2,4,9,11,13,14,16,17,18,19,20,25,27,3
```

In [5]:
```
# Rename some of the features for convenience
features_all = feature_select.rename(columns = {'Facility Name':'facility_name','
                                        'Offers in-center peritoneal dialysis':'per
                                        'Percentage of Medicare patients with Hgb<1
                                        'Medicare_Patient_Hgb>12','Percent of Adult
                                        'HD_Pediatric_Kt/V >= 1.2','Percentage of p
                                        'Percentage of patients with vascular cathe
                                        'Percentage of Adult patients with hypercal
                                        'Percentage of Adult patients with serum ph
                                        'Percentage of Adult patients with serum ph
                                        'Percentage of Adult patients with serum ph
                                        'Percentage of Adult patients with serum ph
                                        'Percentage of Adult patients with serum ph
                                        'Mortality Rate (Facility)':'Mortality_rate
                                        'Hospitalization Rate (Facility)':'Hospital
                                        'Percentage of adult PD patients with Kt/V>
                                        'PD_Pat_Kt/V>=1.7','Standard Infection Rati
```

In [6]:
```
# regrouping the columns to line up the predictor variables together and
# move the facility name to the first column

# To get a list of columns
cols = list(features_all)

# To move the column to last of columns' list
cols.insert(0, cols.pop(cols.index('facility_name')))


# To reorder the columns
features_all = features_all.ix[:, cols]
```

```
C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: Deprec
ationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix (http://
pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix)
  if sys.path[0] == '':
```

In [7]:
```
from sklearn.preprocessing import LabelEncoder

# To convert the categorical inputs of some data fields to numerical nature

numeric = LabelEncoder()
features_all['Profit_YesNo']=numeric.fit_transform(features_all['Profit_YesNo'].a
features_all['chain_owned']=numeric.fit_transform(features_all['chain_owned'].ast
features_all['late_shift']=numeric.fit_transform(features_all['late_shift'].astyp
features_all['hemodial_incenter']=numeric.fit_transform(features_all['hemodial_in
features_all['perit_incenter']=numeric.fit_transform(features_all['perit_incenter
features_all['hemoTrain_home']=numeric.fit_transform(features_all['hemoTrain_home
```

In [8]:
```
# Some of the features contain a "%" symbol, which has to be cleaned.
# Getting rid off the % symbol from the respective columns

features_all[['Medicare_Patient_Hgb<10','Medicare_Patient_Hgb>12','HD_Patient_Kt/
              'Pat_AV_fistulae','Pat_Vasc-Catheter_90days','Pat_hypercalcemia','S
              'SerumPhos_3.5-4.5mg/dL','SerumPhos_4.6-5.5mg/dL','SerumPhos_>7.0mg
= features_all[['Medicare_Patient_Hgb<10','Medicare_Patient_Hgb>12','HD_Patient_K
                'Pat_AV_fistulae','Pat_Vasc-Catheter_90days','Pat_hypercalcemia',
              'SerumPhos_3.5-4.5mg/dL','SerumPhos_4.6-5.5mg/dL','SerumPhos_>7.0mg
              .replace('%','',regex=True).astype('float')
```

Two additional dataset named as "Population_county" and "Income_county" were merged to the original dataset (Dialysis_Mortality) based on State and county data fields, as both could be valuable datafields for our analysis, . The next three cells show the merging and cleaning steps.

In [9]:
```
#Loading county based population size dataset

pop_county= pd.read_csv('Population_county.csv',encoding='latin-1')

# Selecting the total population size as of recent estimate(2016) and for all ages

pop_county = pop_county.loc[(pop_county['YEAR'] == 9) & (pop_county['AGEGRP'] == 

# Slicing the dataset based on County and State, as the county with the same name

pop_county = pop_county.groupby(['County','State'],as_index=False).sum()

# Before merging the datasets, the county names used in both data frames should be
# To fulfill that, all are converted to upper case, and any space within two words

pop_county['County']=pop_county['County'].str.upper()
pop_county['County']=pop_county['County'].str.replace('\s+', '')
features_all['County']=features_all['County'].str.replace('\s+', '')

# Merging the population table with the mortality table based on county and state
mort_pop = pd.merge(features_all, pop_county, how='left',on=['County','State'])

# Moving the county and state to the front columns

c=list(mort_pop)
c.insert(1, c.pop(c.index('County')))
c.insert(2, c.pop(c.index('State')))

mort_pop =mort_pop.ix[:,c]
```

```
C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:29: Deprec
ationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix (http://
pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix)
```

In [10]:
```
#Loading county based income dataset
income_county = pd.read_csv('Income_County.csv',encoding='latin-1')

# Listing the states' name with its abbreviations
state = {'AK': 'Alaska','AL': 'Alabama','AR': 'Arkansas','AS': 'American Samoa','
        'CT': 'Connecticut','DC': 'District of Columbia','DE': 'Delaware','FL': '
        'IA': 'Iowa','ID': 'Idaho','IL': 'Illinois','IN': 'Indiana','KS': 'Kansas
        'MD': 'Maryland','ME': 'Maine','MI': 'Michigan','MN': 'Minnesota','MO': '
        'MT': 'Montana','NA': 'National','NC': 'North Carolina','ND': 'North Dako
        'NJ': 'New Jersey','NM': 'New Mexico','NV': 'Nevada','NY': 'New York','OH
        'PA': 'Pennsylvania','PR': 'Puerto Rico','RI': 'Rhode Island','SC': 'Sout
        'TX': 'Texas','UT': 'Utah','VA': 'Virginia','VI': 'Virgin Islands','VT':
        'WV': 'West Virginia','WY': 'Wyoming'}

# Adding the abbreviations to the datasets
state= {State: abbrev for abbrev, State in state.items()}
income_county['State'] = income_county['State'].map(state)

# Converting the county name into uppercase letters, and any spaces between two w
income_county['County'] = income_county['County'].str.upper()
income_county['County']=income_county['County'].str.replace('\s+', '')

# Merging the income info with the mortality table based on county and state
mort_popIncome = pd.merge(mort_pop, income_county,how='left',on=['County','State'

# Dropping population variable from the dataset, as we already have a population
total_mortRate = mort_popIncome.loc[:,mort_popIncome.columns!='Population']

#converting the features with a string type into the categorical inputs
total_mortRate['County']=numeric.fit_transform(total_mortRate['County'].astype('s
```

```
C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:30: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
```

In [11]:
```
# move the five_star to the last column for convience
co=list(mort_pop)
co.insert(49, co.pop(co.index('five_star')))
total_mortRate =total_mortRate.ix[:,co]
```

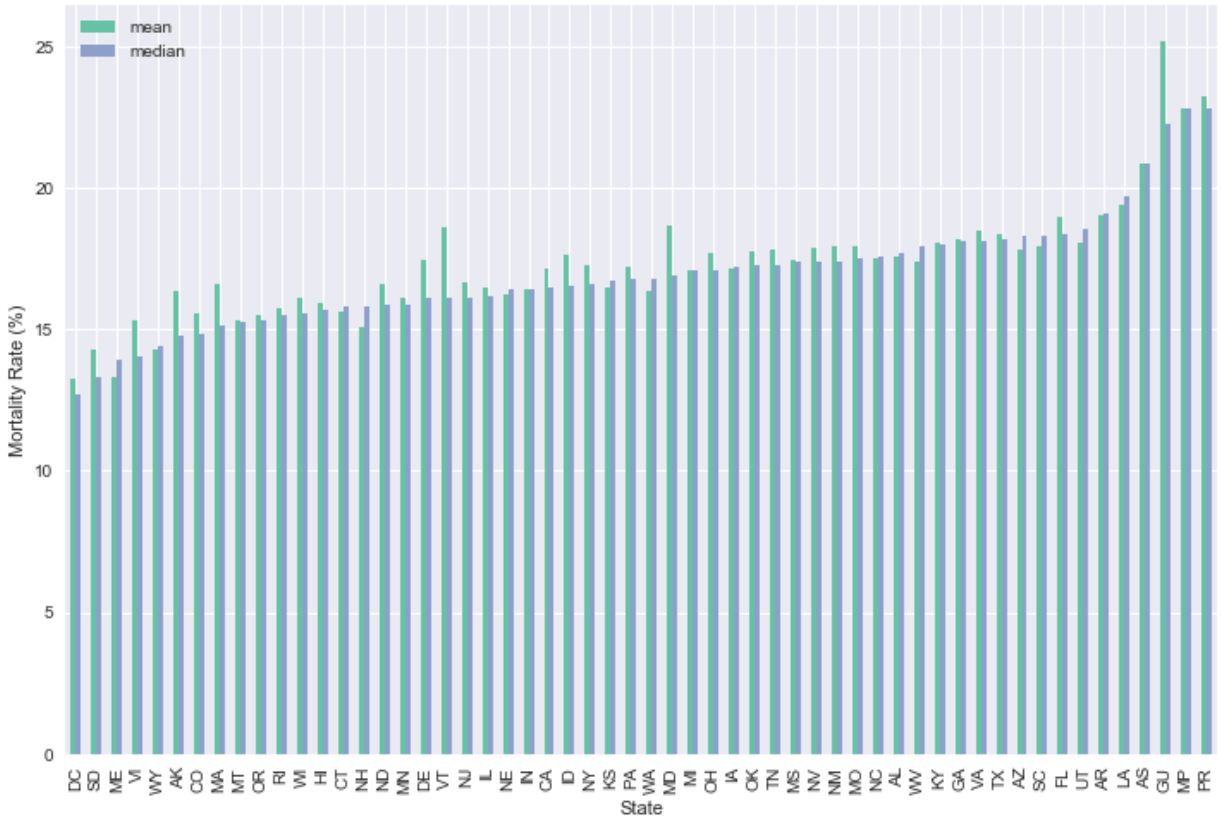
```
C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: Depreca
tionWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix (http://
pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix)
  after removing the cwd from sys.path.
```

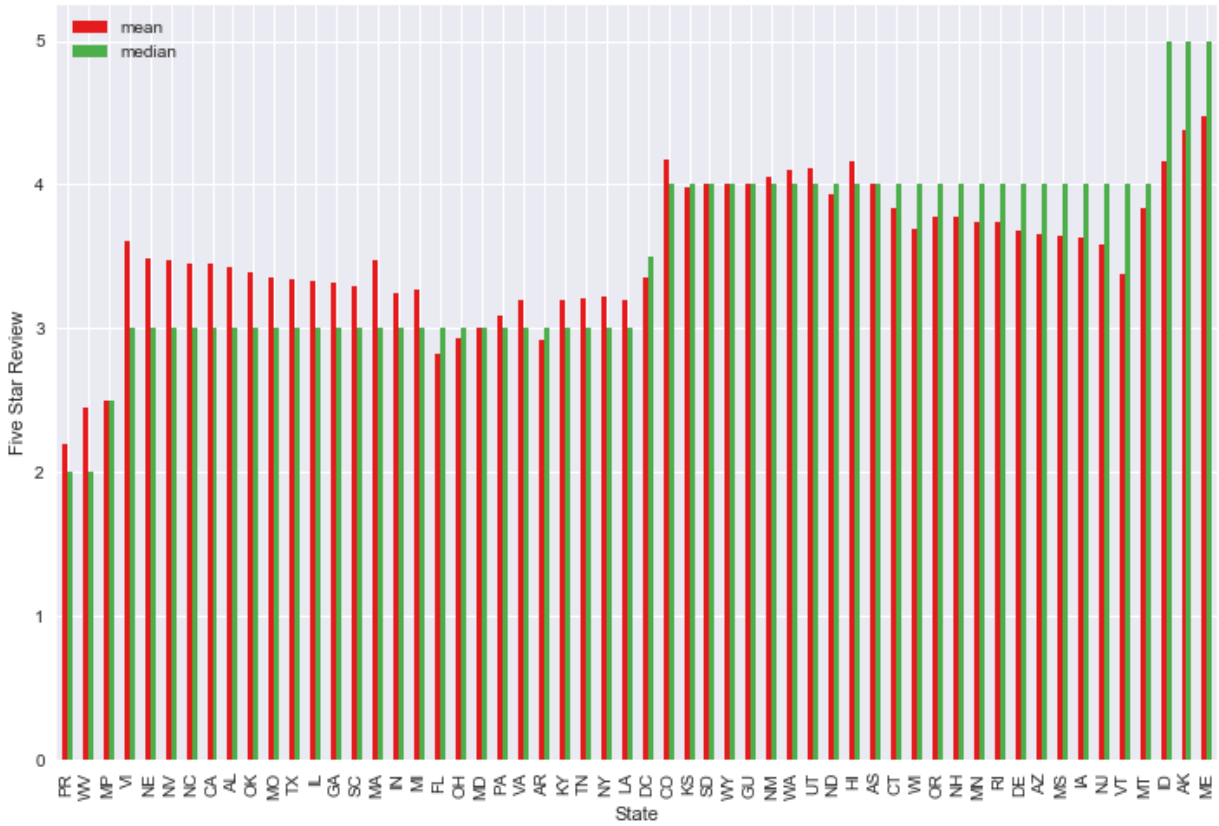
The dataset was cleaned and subset to 6810 observations and 47 data fields, and the dataset is now ready for exploration and data analysis. The mortality rate and five star varibales are plotted against the States to see any trends and variations.

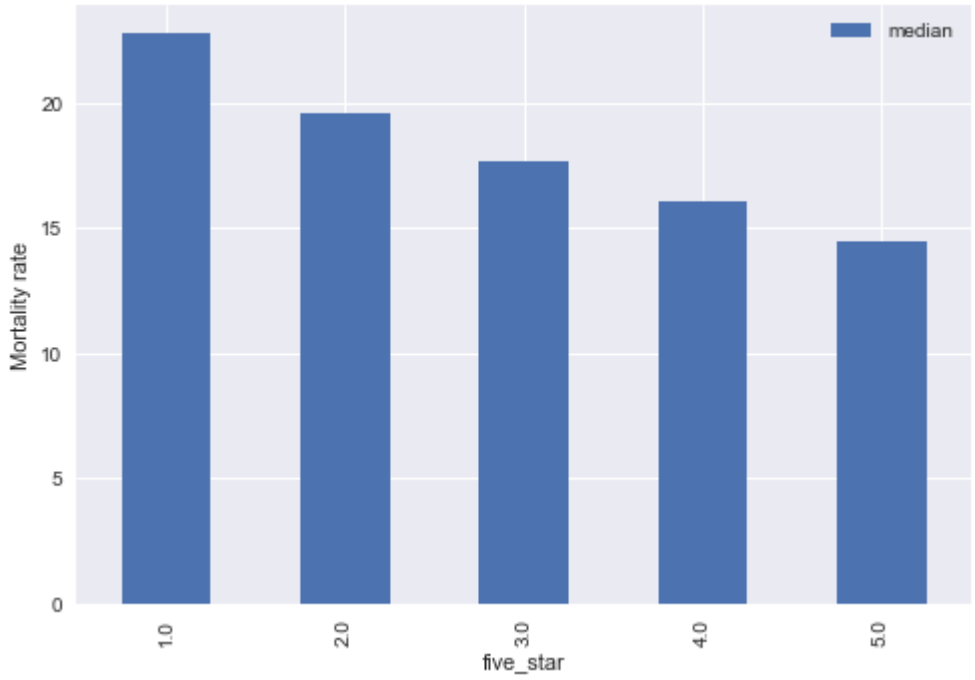
In [12]:
```
# Loading the necessary python packages
from pandas.tools.plotting import scatter_matrix
import seaborn as sns
from scipy.stats import norm
```

In [13]:
```
# Subsetting the State based on the mortality_rate
mort_state = total_mortRate.groupby('State',as_index=True)['Mortality_rate'].agg(
mort_state = mort_state.sort_values(by='mean')
mort_state1 = mort_state.sort_values(by='median')
colors = plt.cm.Set2(np.linspace(0,1,5))
mort_state1.plot.bar(figsize=(12,8),color = colors).set_ylabel('Mortality Rate (%

# Subsetting the State based on the five_star
fiveStar_state = total_mortRate.groupby('State',as_index=True)['five_star'].agg([
fiveStar_state = fiveStar_state.sort_values(by='mean')
fiveStar_state1 = fiveStar_state.sort_values(by='median')
colors = plt.cm.Set1(np.linspace(0,1,5))
fiveStar_state1.plot.bar(figsize=(12,8),color = colors).set_ylabel('Five Star Rev

# Exploring the mortality_rate against the five_star reviews

star = total_mortRate.groupby('five_star',as_index=True)['Mortality_rate'].agg(['
star.plot.bar().set_ylabel('Mortality rate')
```

Out[13]: <matplotlib.text.Text at 0x6f3b79b5c0>



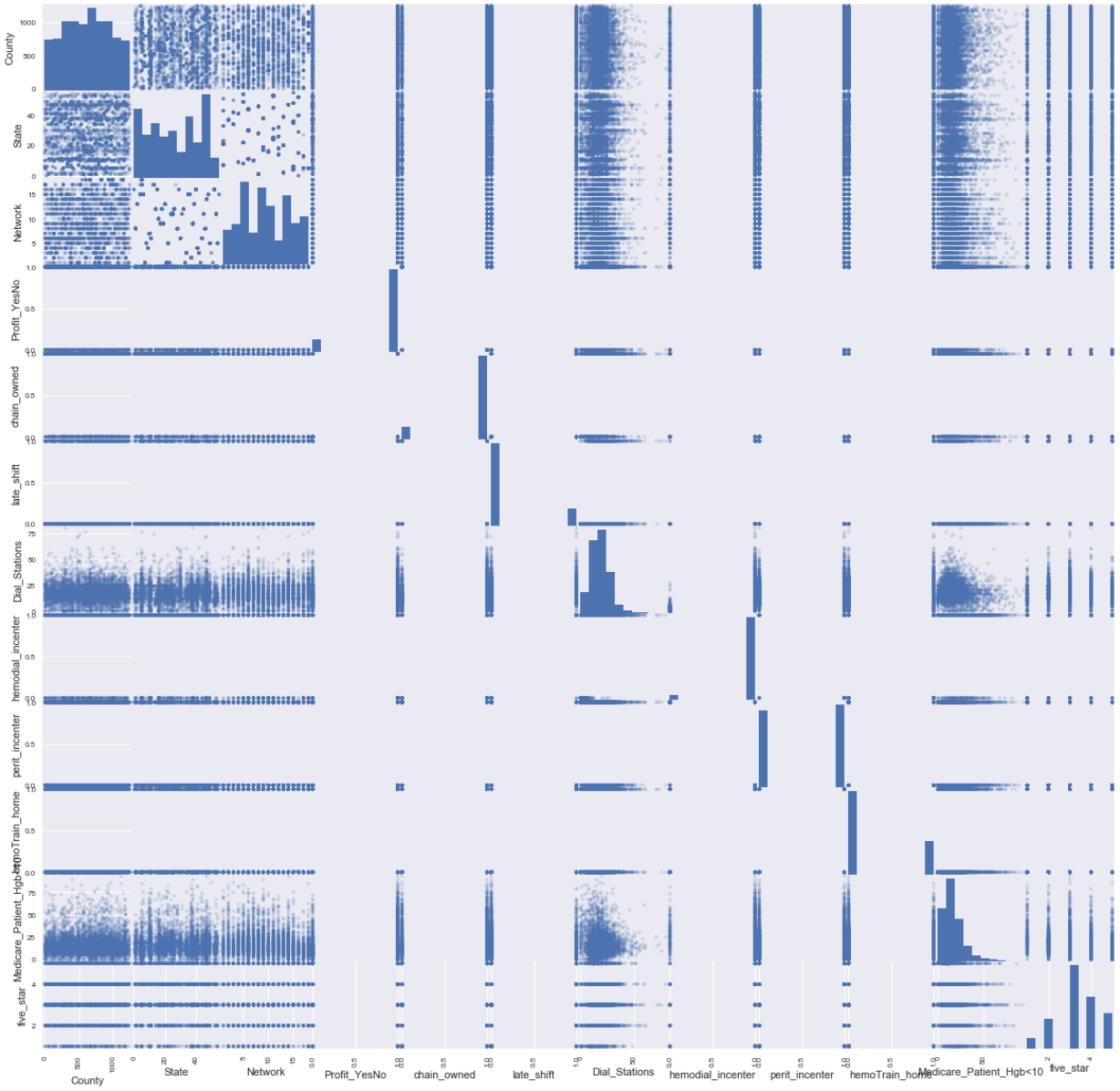


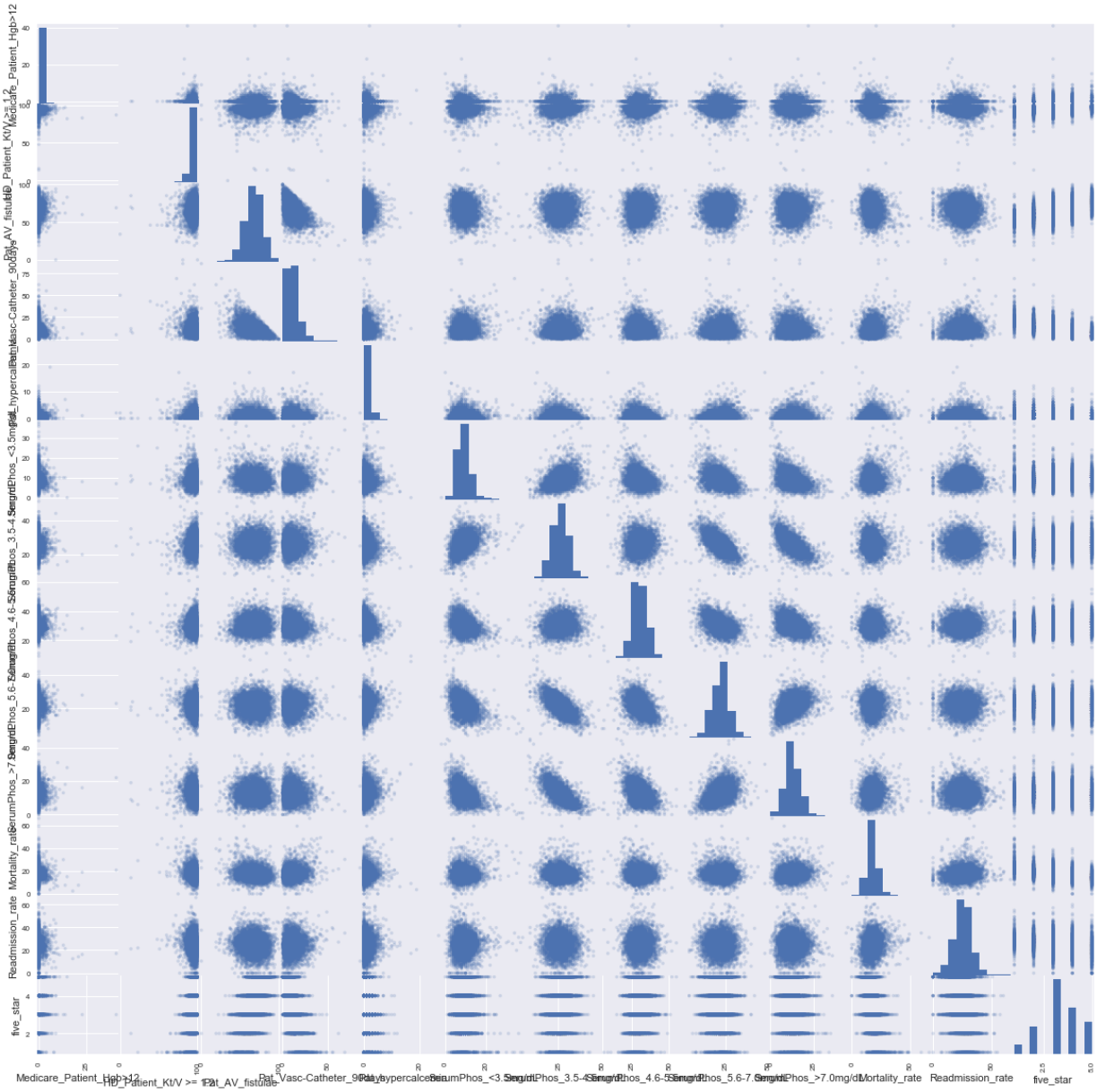


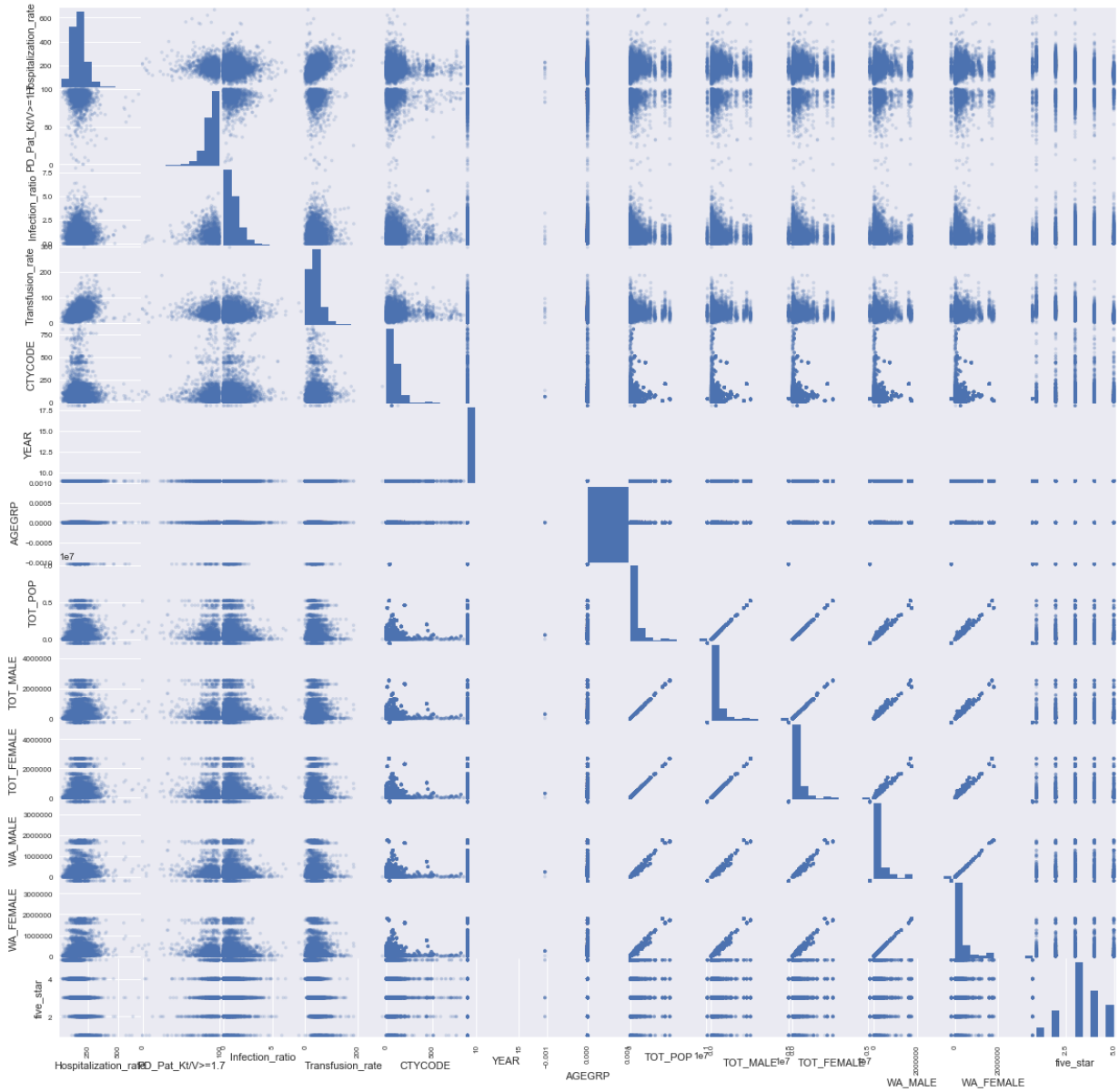
The five star variable was mapped against all the other features to examine any pattern that could be observed. This varibale are expected to show some pattern against the other features.

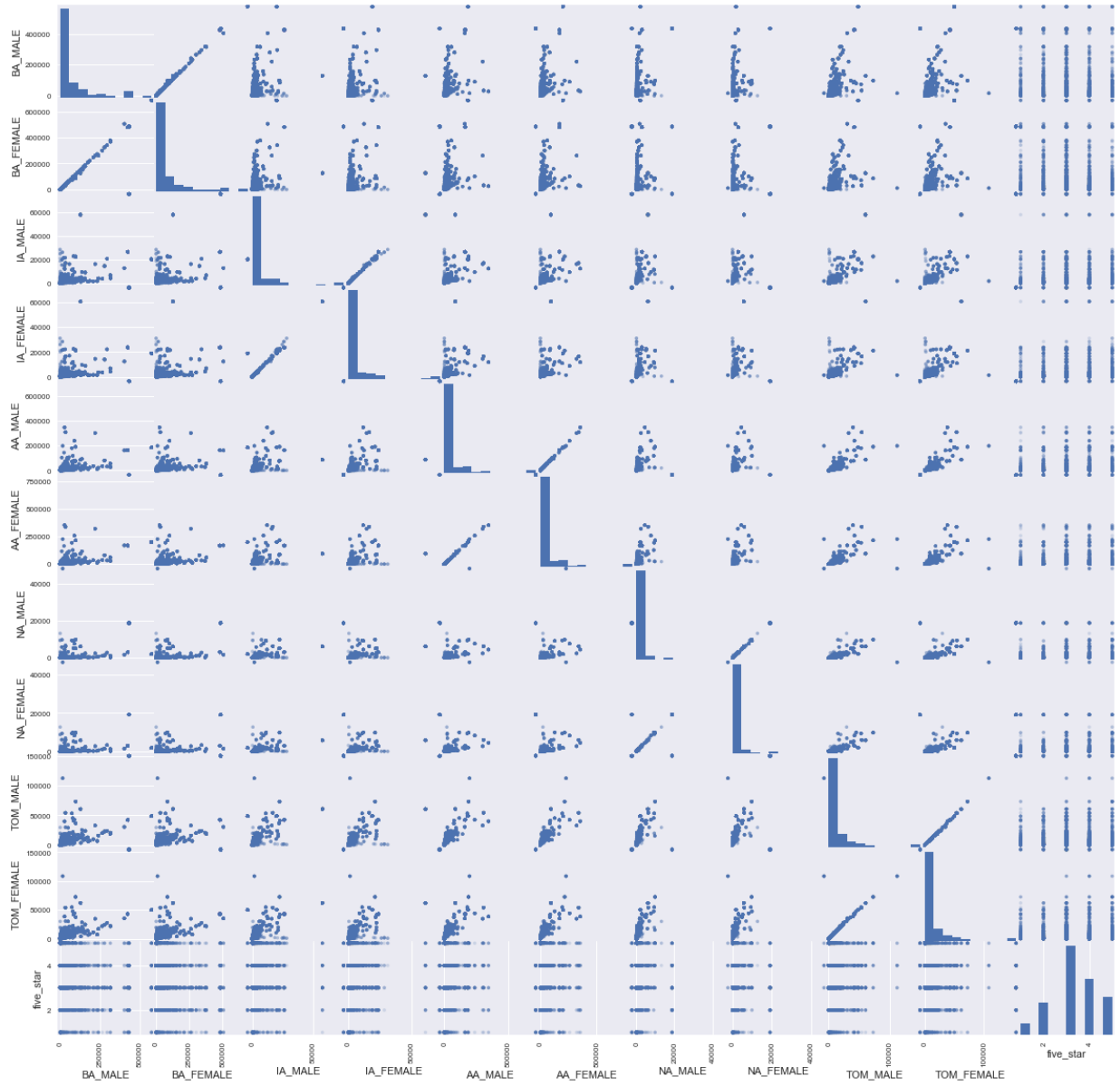
In [14]:
```
# Comparing the trend of the five_star variable to the other features

total_mortRate['State']=numeric.fit_transform(total_mortRate['State'].astype('str
list(total_mortRate).index('five_star')


test1 = total_mortRate.iloc[:, np.r_[1:12,46]]
scatter_matrix(test1, alpha=0.2, figsize=(20, 20), diagonal='hist');

test2 = total_mortRate.iloc[:, np.r_[12:24,46]]
scatter_matrix(test2, alpha=0.2, figsize=(20, 20), diagonal='hist');

test3 = total_mortRate.iloc[:, np.r_[24:36,46]]
scatter_matrix(test3, alpha=0.2, figsize=(20, 20), diagonal='hist');

test4 = total_mortRate.iloc[:, np.r_[36:46,46]]
scatter_matrix(test4, alpha=0.2, figsize=(20, 20), diagonal='hist');

# sns.set(style="whitegrid", color_codes=True)
# sns.regplot(x="five_star", y="Hospitalization_rate", data=total_mortRate);
```

```
C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: FutureW
arning: 'pandas.tools.plotting.scatter_matrix' is deprecated, import 'pandas.pl
otting.scatter_matrix' instead.

C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: Future
Warning: 'pandas.tools.plotting.scatter_matrix' is deprecated, import 'pandas.p
lotting.scatter_matrix' instead.
  # This is added back by InteractiveShellApp.init_path()
C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: Future
Warning: 'pandas.tools.plotting.scatter_matrix' is deprecated, import 'pandas.p
lotting.scatter_matrix' instead.

C:\Users\benbahtin\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:2917: U
serWarning: Attempting to set identical left==right results
in singular transformations; automatically expanding.
left=0.0, right=0.0
  'left=%s, right=%s') % (left, right))
C:\Users\benbahtin\Anaconda3\lib\site-packages\matplotlib\axes\_base.py:3193: U
serWarning: Attempting to set identical bottom==top results
in singular transformations; automatically expanding.
bottom=0.0, top=0.0
  'bottom=%s, top=%s') % (bottom, top))
C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:17: Future
Warning: 'pandas.tools.plotting.scatter_matrix' is deprecated, import 'pandas.p
lotting.scatter_matrix' instead.
```


County
State
Network
Profit_YesNo
chain_owned
late_shift
Dial_Stations
hemodial_incenter
perit_incenter
hemoTrain_home
Medicare_Patient_Hgb<10
five_star


Medicare_Patient_Hgb>12
HD_Patient_Kt/V >= 1.2
Pat_AV_fistula
Pat_Vasc-Catheter_90days
Pat_hypercalcemia
SerumPhos_<3.5mg/dL
SerumPhos_3.5-4.5mg/dL
SerumPhos_4.6-5.5mg/dL
SerumPhos_5.6-7.0mg/dL
SerumPhos_>7.0mg/dL
Mortality_rate
Readmission_rate
five_star

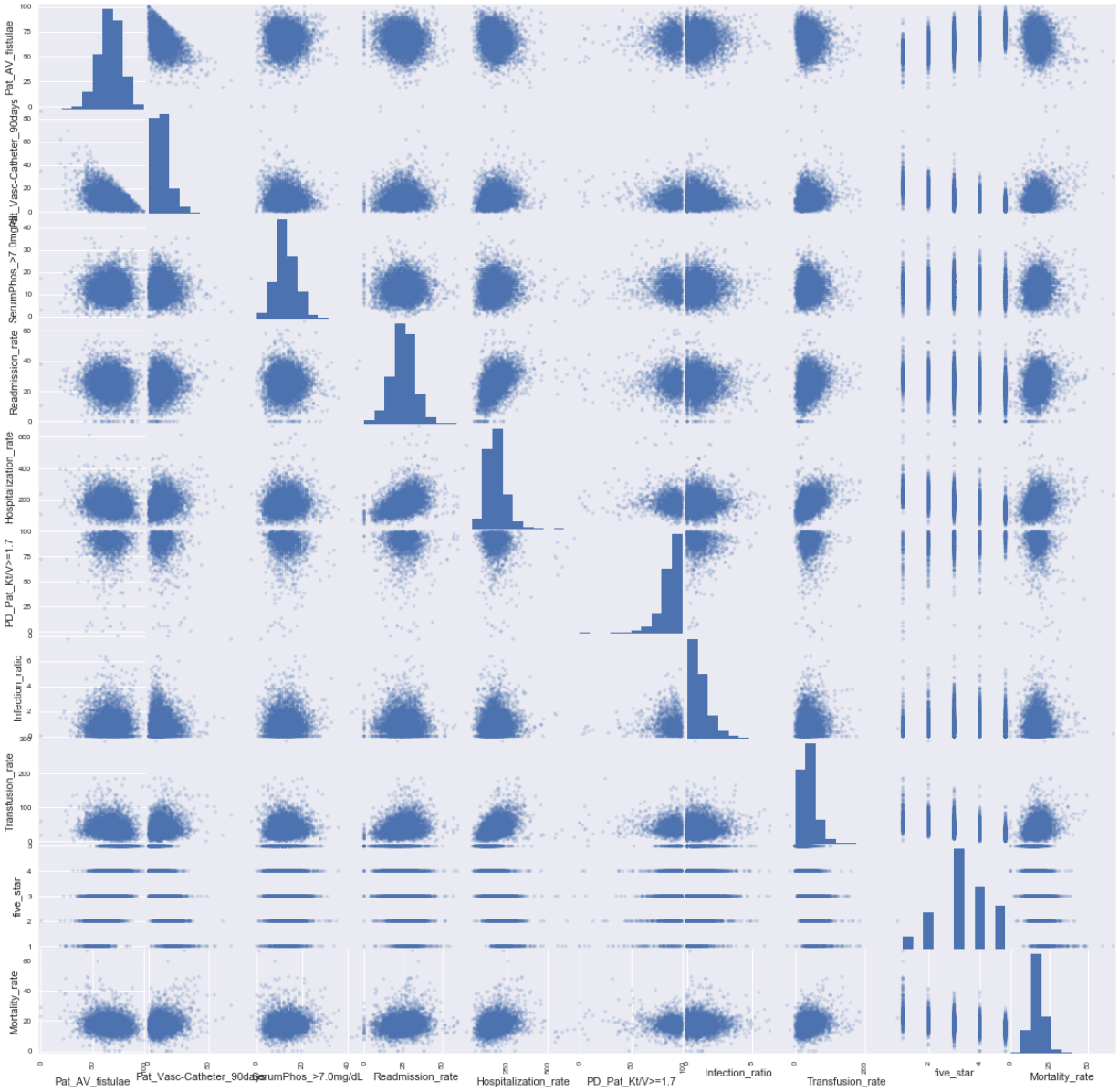

Hospitalization_rate
PD_Pat_Kt/V>=1.7
Infection_ratio
Transfusion_rate
CTYCODE
YEAR
AGEGRP
TOT_POP
TOT_MALE
TOT_FEMALE
WA_MALE
WA_FEMALE
five_star



The mortality rate was also plotted against those features which are believed to show some trend.

In [15]:
```
# Ploting the mortality rate against those features which showes some pattern/trei
relate = total_mortRate.iloc[:, [14,15,21,23,24,25,26,27,46,22]]
relate = scatter_matrix(relate, alpha=0.2, figsize=(20, 20), diagonal='hist');
```

```
C:\Users\benbahtin\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: FutureW
arning: 'pandas.tools.plotting.scatter_matrix' is deprecated, import 'pandas.pl
otting.scatter_matrix' instead.
  This is separate from the ipykernel package so we can avoid doing imports unt
il
```



The above plots shows that the trend of the Mortality_rate or five_star varies from one State to another. Both appeared to classify the States into three to four groups. When both are correlated with the other variables, it can be observed some trends which varies from insignificant to very significant. For example, the mortality rate indicates an increasing trend with hospitalization, readmission and tranfusion rates, while infection ratio seem to show not a strong trend with the mortality rate. However, the other features did not seem to show any significant trend against the mortality rate. The relationship between Mortality-rate and five_star is appeared to be negative (see the histogram). The variations observed across the States can indicate that there could be significant pattern in the facilities given the health data of the patients.

Model Fitting

Loading Libraries

In [16]:
```
# Loading the necessary libraries
from sklearn.preprocessing import Imputer,StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.cross_validation import train_test_split
from sklearn.cluster import AffinityPropagation
from sklearn.cluster import SpectralClustering
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.metrics import silhouette_samples, silhouette_score
```

```
C:\Users\benbahtin\Anaconda3\lib\site-packages\sklearn\cross_validation.py:44:
 DeprecationWarning: This module was deprecated in version 0.18 in favor of the
model_selection module into which all the refactored classes and functions are
 moved. Also note that the interface of the new CV iterators are different from
that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Filling Missing data

In [17]:
```
# Selecting the features

X_col = total_mortRate.columns[1:]
features = total_mortRate[X_col]

# Create our imputer to replace missing values with the median
imp = Imputer(missing_values='NaN', strategy='median', axis=0)
imp_feature = imp.fit_transform(features)

# # Standarizing the dataset
# X_scaled = pd.DataFrame(StandardScaler().fit_transform(imp_feature),columns = X_
# X_scaled.head()
```

Training and Testing dataset

The dataset was randomly split into two groups. K-Means, agglomerative clustering, affinity propagation and DBSCAN were utilized on the first 50% of the dataset. The model with best silhouette score were selected and applied to the rest 50% of the dataset to see the correlation of the results for evaluation purposes.

In [18]:
```
# Splitting the dataset into two, and later to evaluate the clustering centroids j
data_train, data_test= train_test_split(imp_feature, test_size = 0.2, random_stat

print(data_train.shape)
print(data_test.shape)
```

```
(5448, 46)
(1362, 46)
```

Training dataset

K-Means

In [19]:
```
# KMeans
k_range = range(2,11)

for i in k_range:
    model = KMeans(n_clusters=i,random_state=1234)
    pred =model.fit_predict(data_train)
    silhouette_avg = silhouette_score(data_train,pred)
    print('The number of clusters, %d, and silhouette coefficient is %0.2f'% (i,s
```

```
The number of clusters, 2, and silhouette coefficient is 0.85
The number of clusters, 3, and silhouette coefficient is 0.79
The number of clusters, 4, and silhouette coefficient is 0.71
The number of clusters, 5, and silhouette coefficient is 0.65
The number of clusters, 6, and silhouette coefficient is 0.62
The number of clusters, 7, and silhouette coefficient is 0.61
The number of clusters, 8, and silhouette coefficient is 0.62
The number of clusters, 9, and silhouette coefficient is 0.62
The number of clusters, 10, and silhouette coefficient is 0.63
```

Agglomerative clustering

In [20]:
```
# Agglomerative clustering
for j in range(2,11):
    c_agg = AgglomerativeClustering(n_clusters=j)
    c_agg.fit_predict(data_train)
    label_2 = c_agg.labels_
    score_1 = silhouette_score(data_train, label_2, metric='euclidean')
    print("The number of clusters, %d, and silhouette coefficient is %0.2f" % (j,
```

```
The number of clusters, 2, and silhouette coefficient is 0.81
The number of clusters, 3, and silhouette coefficient is 0.80
The number of clusters, 4, and silhouette coefficient is 0.63
The number of clusters, 5, and silhouette coefficient is 0.64
The number of clusters, 6, and silhouette coefficient is 0.60
The number of clusters, 7, and silhouette coefficient is 0.61
The number of clusters, 8, and silhouette coefficient is 0.61
The number of clusters, 9, and silhouette coefficient is 0.61
The number of clusters, 10, and silhouette coefficient is 0.62
```

Affinity propagation

In [21]:
```
# Affinity propagation
c_affPro = AffinityPropagation()
c_affPro.fit_predict(data_train)
cluCentInd = c_affPro.cluster_centers_indices_
labels = c_affPro.labels_
print("Silhouette Coefficient: %0.3f"% silhouette_score(data_train, labels, metri
```

```
Silhouette Coefficient: 0.043
```

DBSCAN

In [22]:
```
# Using PCA for dimensionality reduction
P= PCA(n_components=2)

# adding the PC1 and PC2 to the exsting data frame
data_train = pd.DataFrame(data_train)
data_train['x1']= P.fit_transform(data_train)[:,0]
data_train['y1']= P.fit_transform(data_train)[:,1]

# Applying DBSCAN
A = list(zip(data_train.x1,data_train.y1))
db = DBSCAN().fit(A)
labels = db.labels_
print("Silhouette Coefficient: %0.3f"% silhouette_score(A, labels))
```

```
Silhouette Coefficient: 0.320
```

Based on the silhouette score, the K-Means and agglomerative clustering yielded better performance with the number of clusters used. Reasonable (number of cluster range: 5-10) to strong structure (number of cluster range: 2-4) has been found. However, the DBSCAN and affinity propagation show no substantial to weak structure. As K-Means and agglomerate clustering resulted to almost similar results, one of them, K-Means, with number of clusters of 4 is selected to be utilized on the rest 50% of the data held out. The selection of number of clusters is based on the EDA of five star/mortality rate against the States (see the histograms on EDA section). The States can be categorized into four groups depending on the five star/mortality rate.

Testing dataset

In [23]:
```
# applying Kmeans clustering on the rest 20% of the dataset, testing data
k_range_2 = range(2,11)

for k in k_range_2:
    model = KMeans(n_clusters=k,random_state=1234)
    pred_2 =model.fit_predict(data_test)
    silhouette_avg_2 = silhouette_score(data_test,pred_2)
    print('The number of clusters, %d, and silhouette coefficient is %0.2f'% (k,s
```

```
The number of clusters, 2, and silhouette coefficient is 0.85
The number of clusters, 3, and silhouette coefficient is 0.81
The number of clusters, 4, and silhouette coefficient is 0.70
The number of clusters, 5, and silhouette coefficient is 0.65
The number of clusters, 6, and silhouette coefficient is 0.61
The number of clusters, 7, and silhouette coefficient is 0.60
The number of clusters, 8, and silhouette coefficient is 0.61
The number of clusters, 9, and silhouette coefficient is 0.62
The number of clusters, 10, and silhouette coefficient is 0.62
```

Plotting the clusters for training and testing dataset

In [24]:
```
# Reducing the number of features (dimentionality reduction for visualization pur
dim_redu_train =P.fit_transform(data_train)

# fit to the kmeans model
km =  KMeans(n_clusters=4)
pred_train = km.fit_predict(dim_redu_train)

# Adding the cluster values to the dataframe, data_1
data_train['pred_cluster']= pred_train


# converting cluster into str nature for the plotting purposes
data_train['pred_cluster'] = data_train['pred_cluster'].astype('str')

print(pd.DataFrame(data_train.pred_cluster.value_counts()))
data_train.head()
```

```
   pred_cluster
0          4212
3           785
1           312
2           139
```

Out[24]:

	0	1	2	3	4	5	6	7	8	9	...	39	40	41	42	43
0	1208.0	16.0	10.0	1.0	1.0	0.0	29.0	1.0	1.0	0.0	...	18920.0	20620.0	158.0	170.0	6462.0
1	393.0	16.0	10.0	1.0	1.0	0.0	8.0	1.0	1.0	0.0	...	33.0	43.0	2.0	3.0	136.0
2	326.0	2.0	13.0	1.0	1.0	0.0	11.0	1.0	0.0	0.0	...	38.0	27.0	8.0	8.0	62.0
3	526.0	49.0	5.0	1.0	1.0	0.0	16.0	1.0	1.0	0.0	...	111.0	173.0	27.0	30.0	395.0
4	966.0	26.0	12.0	1.0	1.0	0.0	14.0	1.0	1.0	1.0	...	5619.0	6412.0	203.0	193.0	4687.0

5 rows × 49 columns

In [25]:
```
# Reducing the number of features (dimentionality reduction for visualization pur

dim_redu_test =P.fit_transform(data_test)

# adding the PC1 and PC2 to the exsting data frame

data_test = pd.DataFrame(data_test)
data_test['x1']= P.fit_transform(data_test)[:,0]
data_test['y1']= P.fit_transform(data_test)[:,1]

# fit to the kmeans model
km =  KMeans(n_clusters=4)
pred_test = km.fit_predict(dim_redu_test)

# Adding the cluster values to the dataframe, data_test
data_test['pred_cluster']= pred_test


# converting cluster into str nature for the plotting purposes
data_test['pred_cluster'] = data_test['pred_cluster'].astype('str')

print(pd.DataFrame(data_test.pred_cluster.value_counts()))
data_test.head()
```

```
   pred_cluster
0          1059
3           199
2            72
1            32
```

Out[25]:

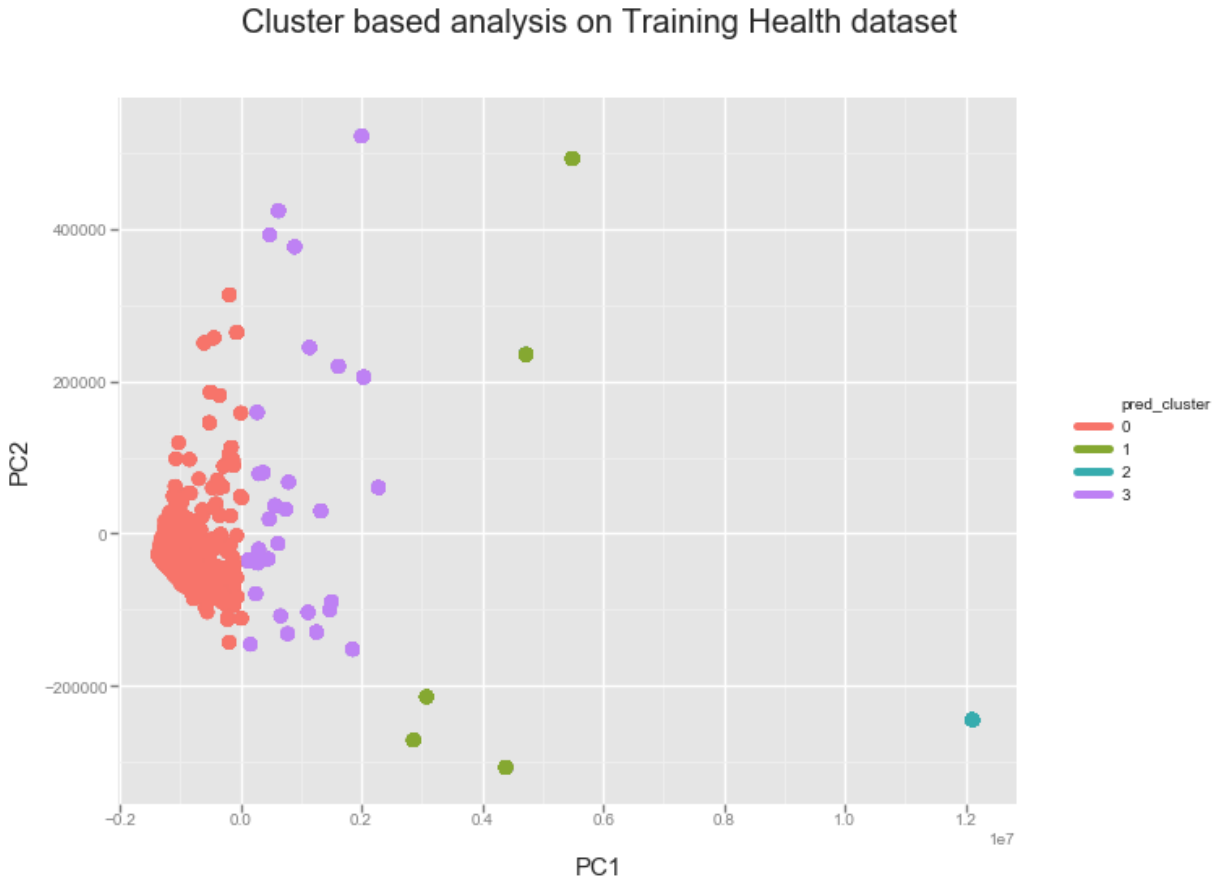
	0	1	2	3	4	5	6	7	8	9	...	39	40	41	42	43
0	838.0	24.0	11.0	1.0	1.0	0.0	18.0	1.0	1.0	1.0	...	3779.0	4499.0	68.0	69.0	2659.0
1	675.0	25.0	11.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	...	587.0	474.0	13.0	11.0	234.0
2	167.0	1.0	8.0	1.0	0.0	0.0	25.0	1.0	1.0	1.0	...	426.0	628.0	64.0	49.0	1058.0
3	1054.0	52.0	16.0	1.0	1.0	1.0	20.0	1.0	0.0	0.0	...	5172.0	6589.0	1292.0	1442.0	9693.0
4	679.0	19.0	9.0	1.0	1.0	1.0	20.0	1.0	1.0	1.0	...	488.0	557.0	28.0	23.0	868.0

5 rows × 49 columns

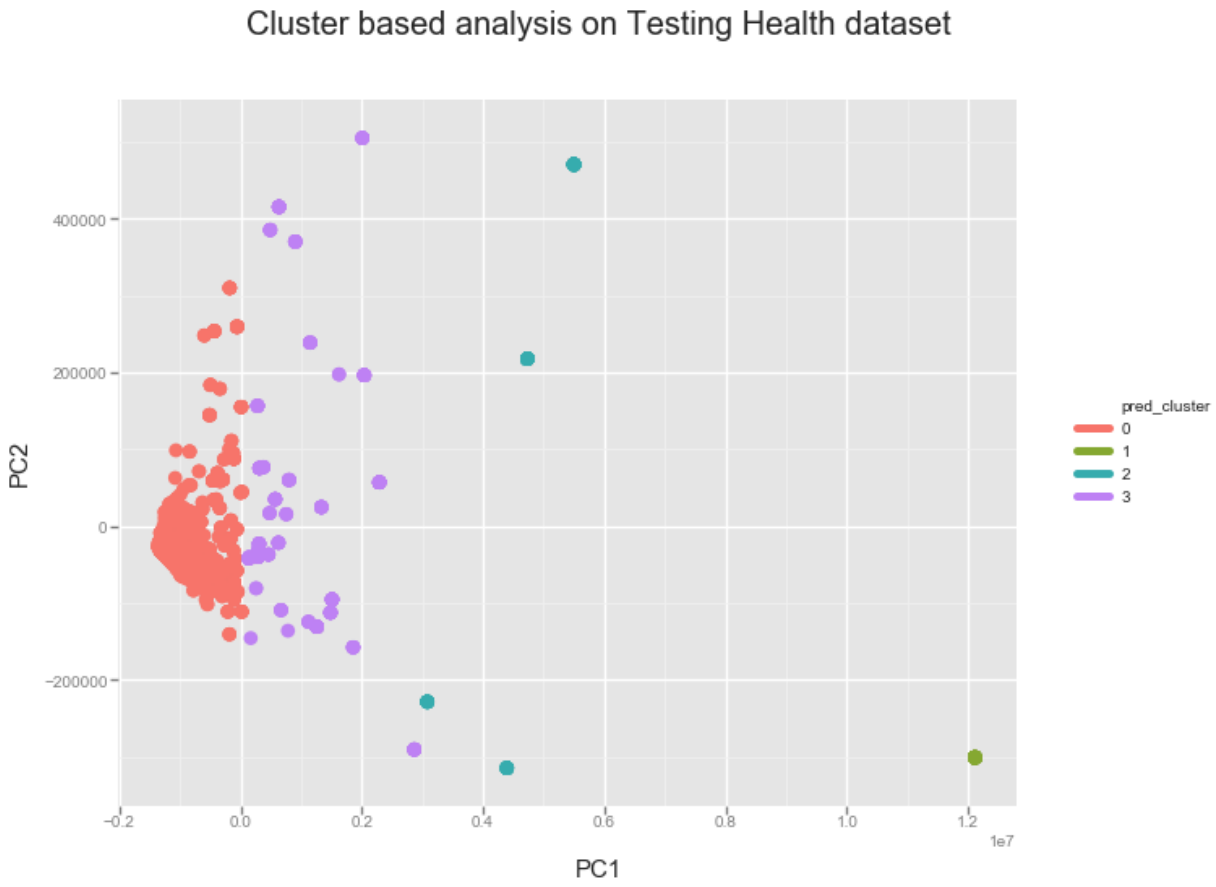
In [26]:
```
! pip install ggplot
from ggplot import *
```

```
Requirement already satisfied: ggplot in c:\users\benbahtin\anaconda3\lib\site-
packages
Requirement already satisfied: matplotlib in c:\users\benbahtin\anaconda3\lib\s
ite-packages (from ggplot)
Requirement already satisfied: patsy>=0.4 in c:\users\benbahtin\anaconda3\lib\s
ite-packages (from ggplot)
Requirement already satisfied: six in c:\users\benbahtin\anaconda3\lib\site-pac
kages (from ggplot)
Requirement already satisfied: pandas in c:\users\benbahtin\anaconda3\lib\site-
packages (from ggplot)
Requirement already satisfied: scipy in c:\users\benbahtin\anaconda3\lib\site-p
ackages (from ggplot)
Requirement already satisfied: brewer2mpl in c:\users\benbahtin\anaconda3\lib\s
ite-packages (from ggplot)
Requirement already satisfied: statsmodels in c:\users\benbahtin\anaconda3\lib
\site-packages (from ggplot)
Requirement already satisfied: cycler in c:\users\benbahtin\anaconda3\lib\site-
packages (from ggplot)
Requirement already satisfied: numpy in c:\users\benbahtin\anaconda3\lib\site-p
ackages (from ggplot)
Requirement already satisfied: python-dateutil in c:\users\benbahtin\anaconda3
\lib\site-packages (from matplotlib->ggplot)
Requirement already satisfied: pytz in c:\users\benbahtin\anaconda3\lib\site-pa
ckages (from matplotlib->ggplot)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=1.5.6 in c:\u
sers\benbahtin\anaconda3\lib\site-packages (from matplotlib->ggplot)

C:\Users\benbahtin\Anaconda3\lib\site-packages\ggplot\utils.py:81: FutureWarnin
g: pandas.tslib is deprecated and will be removed in a future version.
You can access Timestamp as pandas.Timestamp
  pd.tslib.Timestamp,
C:\Users\benbahtin\Anaconda3\lib\site-packages\ggplot\stats\smoothers.py:4: Fut
ureWarning: The pandas.lib module is deprecated and will be removed in a future
version. These are private functions and can be accessed from pandas._libs.lib
 instead
  from pandas.lib import Timestamp
C:\Users\benbahtin\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56:
FutureWarning: The pandas.core.datetools module is deprecated and will be remov
ed in a future version. Please use the pandas.tseries module instead.
  from pandas.core import datetools
```

In [27]:
```
# For the training dataset
a =ggplot(data_train,aes('x1','y1',color='pred_cluster')) + geom_point(size=75) +
       xlab(element_text(size=15, text='PC1')) + ylab(element_text(size=15, text=

# For the testing dataset
b = ggplot(data_test,aes('x1','y1',color='pred_cluster')) + geom_point(size=75) +
       xlab(element_text(size=15, text='PC1')) + ylab(element_text(size=15, text=

print(a)
b
```



<ggplot: (-9223372006969655499)>



Out[27]: <ggplot: (-9223372006969655534)>

Subsetting the members of the clusters based on States

In [28]:
```
# Assigning the existing names to the column index
AllColumns = list(X_col) + list(['x1','y1','pred_cluster'])
data_train.columns = AllColumns
data_test.columns = AllColumns
```

In [29]:
```
state = {0:'AK',1:'AL',2:'AR',3:'AS',4:'AZ',5:'CA',6:'CO',7:'CT',8:'DC',9:'DE',10
         12:'GU',13:'HI',14:'IA',15:'ID',16:'IL',17:'IN',18:'KS',19:'KY',20:'LA',
         22:'MD',23:'ME',24:'MI',25:'MN',26:'MO',27:'MP',28:'MS',29:'MT',30:'NC',
         33:'NH',34:'NJ',35:'NM',36:'NV',37:'NY',38:'OH',39:'OK',40:'OR',41:'PA',
         44:'SC',45:'SD',46:'TN',47:'TX',48:'UT',
         49:'VA',50:'VI',51:'VT',52:'WA',53:'WI',54:'WV',55:'WY'}

# data_train['State'] = data_train['State'].astype(str)
data_train.State= data_train.State.replace(state)
```

In [30]:
```
data_train['State'] = data_train['State'].astype('str')

# let see which State goes to each cluster
data_train['pred_cluster'] = data_train['pred_cluster'].astype('int')
data_train['0'] = data_train['pred_cluster'] == 0
data_train['1'] = data_train['pred_cluster'] == 1
data_train['2'] = data_train['pred_cluster'] == 2
data_train['3'] = data_train['pred_cluster'] == 3

cluster_0 = data_train.groupby('0')['State'].value_counts().reset_index(name="cou
cluster_1 = data_train.groupby('1')['State'].value_counts().reset_index(name="cou
cluster_2 = data_train.groupby('2')['State'].value_counts().reset_index(name="cou
cluster_3 = data_train.groupby('3')['State'].value_counts().reset_index(name="cou

# w.to_csv('cluster-0.csv')

# Showing the cluster-0 data which include the State, counts of the dialysis faci
cluster_0 = cluster_0[cluster_0['0'] == True]
cluster_0
```

Out[30]:

	0	State	count
14	True	GA	281
15	True	TX	260
16	True	FL	199
17	True	OH	197
18	True	PA	162
19	True	NC	159
20	True	TN	147
21	True	CA	145
22	True	IL	140
23	True	VA	140
24	True	IN	137
25	True	AL	136
26	True	MD	136
27	True	LA	130
28	True	MO	129
29	True	NJ	129
30	True	SC	117
31	True	MI	115
32	True	NY	99
33	True	KY	93
34	True	WI	87
35	True	MS	78

	0	State	count
36	True	MN	75
37	True	OK	65
38	True	CO	61
39	True	WA	60
40	True	AR	58
41	True	MA	57
42	True	OR	54
43	True	IA	53
44	True	KS	46
45	True	NV	40
46	True	AZ	37
47	True	CT	36
48	True	PR	36
49	True	NM	34
50	True	WV	34
51	True	NE	28
52	True	ID	26
53	True	UT	22
54	True	DE	20
55	True	DC	19
56	True	HI	19
57	True	SD	19
58	True	ME	15
59	True	ND	13
60	True	NH	13
61	True	RI	11
62	True	MT	10
63	True	AK	9
64	True	VT	7
65	True	WY	7
66	True	VI	5
67	True	GU	4
68	True	MP	2
69	True	AS	1

In [31]:
```
# Showing the cluster-1 data
cluster_1 = cluster_1[cluster_1['1'] == True]
cluster_1
```

Out[31]:

	1	State	count
56	True	IL	96
57	True	TX	94
58	True	CA	64
59	True	AZ	58

In [32]:
```
# Showing the cluster-2 data
cluster_2 = cluster_2[cluster_2['2'] == True]
cluster_2
```

Out[32]:

	2	State	count
56	True	CA	139

In [33]:
```
# Showing the cluster-3 data
cluster_3 = cluster_3[cluster_3['3'] == True]
cluster_3
```

Out[33]:

	3	State	count
56	True	FL	155
57	True	CA	141
58	True	TX	134
59	True	NY	111
60	True	PA	71
61	True	MI	55
62	True	OH	53
63	True	MN	19
64	True	WA	14
65	True	MA	13
66	True	UT	10
67	True	VA	9