

Project1-EDA

January 27, 2020

1 Project 1 - Clustering

1.0.1 Charlie Marshall

1.0.2 Professor Klabjan

1.0.3 IEMS 308

1.0.4 27 January 2019

```
In [1]: ##### Loading significant packages #####
```

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
from scipy import stats
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.metrics import silhouette_score, silhouette_samples
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
In [2]: ##### Loading Medicare Provider Payment Data #####
```

```
df = pd.read_csv("Medicare_Provider_Util_Payment_PUF_CY2017/Medicare_Provider_Util_Paym
df = df.drop([0])      # Dropping copyright statement
df.head()
```

```
/Users/charlesmarshall/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshell.py:
interactivity=interactivity, compiler=compiler, result=result)
```

```
Out[2]:      npi npes_provider_last_org_name npes_provider_first_name \
1  1003000126      ENKESHAFI      ARDALAN
2  1003000126      ENKESHAFI      ARDALAN
3  1003000126      ENKESHAFI      ARDALAN
4  1003000126      ENKESHAFI      ARDALAN
5  1003000126      ENKESHAFI      ARDALAN
```

	nppes_provider_mi	nppes_credentials	nppes_provider_gender	nppes_entity_code	\
1	NaN		M.D.	M	I
2	NaN		M.D.	M	I
3	NaN		M.D.	M	I
4	NaN		M.D.	M	I
5	NaN		M.D.	M	I

	nppes_provider_street1	nppes_provider_street2	nppes_provider_city	...	\
1	900 SETON DR	NaN	CUMBERLAND	...	
2	900 SETON DR	NaN	CUMBERLAND	...	
3	900 SETON DR	NaN	CUMBERLAND	...	
4	900 SETON DR	NaN	CUMBERLAND	...	
5	900 SETON DR	NaN	CUMBERLAND	...	

	hcpcs_code	hcpcs_description	\
1	99217	Hospital observation care discharge	
2	99218	Hospital observation care typically 30 minutes	
3	99219	Hospital observation care typically 50 minutes	
4	99220	Hospital observation care typically 70 minutes...	
5	99221	Initial hospital inpatient care, typically 30 ...	

	hcpcs_drug_indicator	line_srvc_cnt	bene_unique_cnt	bene_day_srvc_cnt	\
1	N	100.0	96.0	100.0	
2	N	26.0	25.0	26.0	
3	N	52.0	51.0	52.0	
4	N	59.0	59.0	59.0	
5	N	16.0	16.0	16.0	

	average_Medicare_allowed_amt	average_submitted_chrg_amt	\
1	73.398800	325.780000	
2	100.080000	449.000000	
3	136.380000	614.000000	
4	190.363729	755.932203	
5	101.680000	462.812500	

	average_Medicare_payment_amt	average_Medicare_standard_amt
1	56.827200	57.492400
2	78.460000	79.306154
3	102.807692	103.895385
4	141.293559	142.865763
5	79.710000	80.750000

[5 rows x 26 columns]

1.1 Removing Outliers of Numerical Data

All data points outside 3 standard deviations from the mean are removed

```

In [3]: # Removes all outliers from numerical columns
        # Constrains will contain `True` or `False` depending on if it is a value below the threshold
        # Drop (inplace) values set to be rejected
        def drop_numerical_outliers(df, z_thresh=3):
            constrains = df.select_dtypes(include=[np.number]).apply(lambda x: np.abs(stats.zscore(x)) > z_thresh)
            df.drop(df.index[~constrains], inplace=True)

        drop_numerical_outliers(df)

```

1.2 Start Explanatory Data Analysis

```
In [4]: df.shape
```

```
Out[4]: (9689357, 26)
```

```
In [5]: df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 9689357 entries, 1 to 9847443
Data columns (total 26 columns):
npi                                int64
nppes_provider_last_org_name       object
nppes_provider_first_name          object
nppes_provider_mi                  object
nppes_credentials                  object
nppes_provider_gender              object
nppes_entity_code                  object
nppes_provider_street1             object
nppes_provider_street2            object
nppes_provider_city                object
nppes_provider_zip                 object
nppes_provider_state               object
nppes_provider_country             object
provider_type                      object
medicare_participation_indicator    object
place_of_service                   object
hcpcs_code                         object
hcpcs_description                  object
hcpcs_drug_indicator              object
line_srvc_cnt                      float64
bene_unique_cnt                   float64
bene_day_srvc_cnt                  float64
average_Medicare_allowed_amt       float64
average_submitted_chrg_amt         float64
average_Medicare_payment_amt       float64
average_Medicare_standard_amt      float64
dtypes: float64(7), int64(1), object(18)
memory usage: 1.9+ GB

```

```
In [6]: df.describe()
```

```
Out [6]:
```

	npi	line_srvc_cnt	bene_unique_cnt	bene_day_srvc_cnt \
count	9.689357e+06	9.689357e+06	9.689357e+06	9.689357e+06
mean	1.499871e+09	1.606119e+02	7.431467e+01	1.183432e+02
std	2.877043e+08	5.503218e+02	1.361547e+02	2.506905e+02
min	1.003000e+09	7.500000e+00	1.100000e+01	1.100000e+01
25%	1.255302e+09	2.100000e+01	1.700000e+01	2.000000e+01
50%	1.497987e+09	4.300000e+01	3.300000e+01	4.100000e+01
75%	1.740680e+09	1.180000e+02	7.600000e+01	1.070000e+02
max	1.993000e+09	1.561200e+04	3.590000e+03	6.096000e+03

	average_Medicare_allowed_amt	average_submitted_chrg_amt \
count	9.689357e+06	9.689357e+06
mean	8.450123e+01	2.701615e+02
std	9.290735e+01	4.035712e+02
min	6.035380e-05	6.035380e-05
25%	2.411000e+01	5.700000e+01
50%	6.364000e+01	1.436972e+02
75%	1.114200e+02	2.880000e+02
max	9.309430e+02	3.544182e+03

	average_Medicare_payment_amt	average_Medicare_standard_amt
count	9.689357e+06	9.689357e+06
mean	6.422428e+01	6.483764e+01
std	7.217548e+01	7.248741e+01
min	0.000000e+00	0.000000e+00
25%	1.908958e+01	1.988481e+01
50%	4.633478e+01	4.719936e+01
75%	8.317672e+01	8.334863e+01
max	7.234023e+02	7.258472e+02

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

```
Out [7]:
```

	npi	line_srvc_cnt	bene_unique_cnt \
npi	1.000000	0.000278	0.000329
line_srvc_cnt	0.000278	1.000000	0.404730
bene_unique_cnt	0.000329	0.404730	1.000000
bene_day_srvc_cnt	0.000358	0.513464	0.814087
average_Medicare_allowed_amt	-0.000769	-0.083963	-0.045585
average_submitted_chrg_amt	-0.000168	-0.081258	-0.059439
average_Medicare_payment_amt	-0.000711	-0.083876	-0.047775
average_Medicare_standard_amt	-0.000728	-0.084124	-0.047955

	bene_day_srvc_cnt \
npi	0.000358
line_srvc_cnt	0.513464
bene_unique_cnt	0.814087

bene_day_srvc_cnt	1.000000
average_Medicare_allowed_amt	-0.066698
average_submitted_chrg_amt	-0.086088
average_Medicare_payment_amt	-0.069676
average_Medicare_standard_amt	-0.069581

	average_Medicare_allowed_amt \
npi	-0.000769
line_srvc_cnt	-0.083963
bene_unique_cnt	-0.045585
bene_day_srvc_cnt	-0.066698
average_Medicare_allowed_amt	1.000000
average_submitted_chrg_amt	0.718817
average_Medicare_payment_amt	0.992497
average_Medicare_standard_amt	0.988778

	average_submitted_chrg_amt \
npi	-0.000168
line_srvc_cnt	-0.081258
bene_unique_cnt	-0.059439
bene_day_srvc_cnt	-0.086088
average_Medicare_allowed_amt	0.718817
average_submitted_chrg_amt	1.000000
average_Medicare_payment_amt	0.715348
average_Medicare_standard_amt	0.708221

	average_Medicare_payment_amt \
npi	-0.000711
line_srvc_cnt	-0.083876
bene_unique_cnt	-0.047775
bene_day_srvc_cnt	-0.069676
average_Medicare_allowed_amt	0.992497
average_submitted_chrg_amt	0.715348
average_Medicare_payment_amt	1.000000
average_Medicare_standard_amt	0.992823

	average_Medicare_standard_amt
npi	-0.000728
line_srvc_cnt	-0.084124
bene_unique_cnt	-0.047955
bene_day_srvc_cnt	-0.069581
average_Medicare_allowed_amt	0.988778
average_submitted_chrg_amt	0.708221
average_Medicare_payment_amt	0.992823
average_Medicare_standard_amt	1.000000

1.3 EDA of Numerical Data

Starting with EDA with an analysis of the numerical features in the data, with a goal of understanding the distributions of the data.

```
In [8]: df_num = df.select_dtypes(include = ['float64'])
        df_num.info()
        df_num.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9689357 entries, 1 to 9847443
Data columns (total 7 columns):
line_srvc_cnt          float64
bene_unique_cnt        float64
bene_day_srvc_cnt      float64
average_Medicare_allowed_amt  float64
average_submitted_chrg_amt  float64
average_Medicare_payment_amt  float64
average_Medicare_standard_amt float64
dtypes: float64(7)
memory usage: 591.4 MB
```

```
Out [8]:
```

	line_srvc_cnt	bene_unique_cnt	bene_day_srvc_cnt	\
count	9.689357e+06	9.689357e+06	9.689357e+06	
mean	1.606119e+02	7.431467e+01	1.183432e+02	
std	5.503218e+02	1.361547e+02	2.506905e+02	
min	7.500000e+00	1.100000e+01	1.100000e+01	
25%	2.100000e+01	1.700000e+01	2.000000e+01	
50%	4.300000e+01	3.300000e+01	4.100000e+01	
75%	1.180000e+02	7.600000e+01	1.070000e+02	
max	1.561200e+04	3.590000e+03	6.096000e+03	

	average_Medicare_allowed_amt	average_submitted_chrg_amt	\
count	9.689357e+06	9.689357e+06	
mean	8.450123e+01	2.701615e+02	
std	9.290735e+01	4.035712e+02	
min	6.035380e-05	6.035380e-05	
25%	2.411000e+01	5.700000e+01	
50%	6.364000e+01	1.436972e+02	
75%	1.114200e+02	2.880000e+02	
max	9.309430e+02	3.544182e+03	

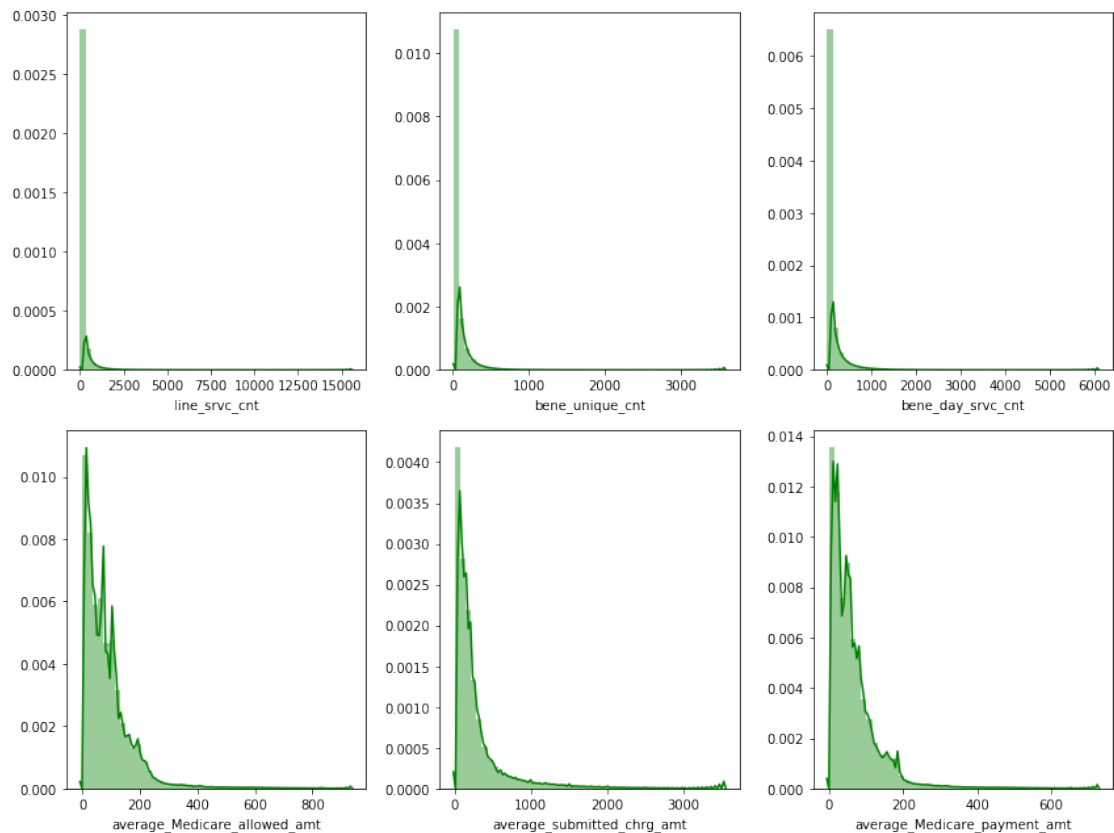
	average_Medicare_payment_amt	average_Medicare_standard_amt
count	9.689357e+06	9.689357e+06
mean	6.422428e+01	6.483764e+01
std	7.217548e+01	7.248741e+01
min	0.000000e+00	0.000000e+00
25%	1.908958e+01	1.988481e+01

50%	4.633478e+01	4.719936e+01
75%	8.317672e+01	8.334863e+01
max	7.234023e+02	7.258472e+02

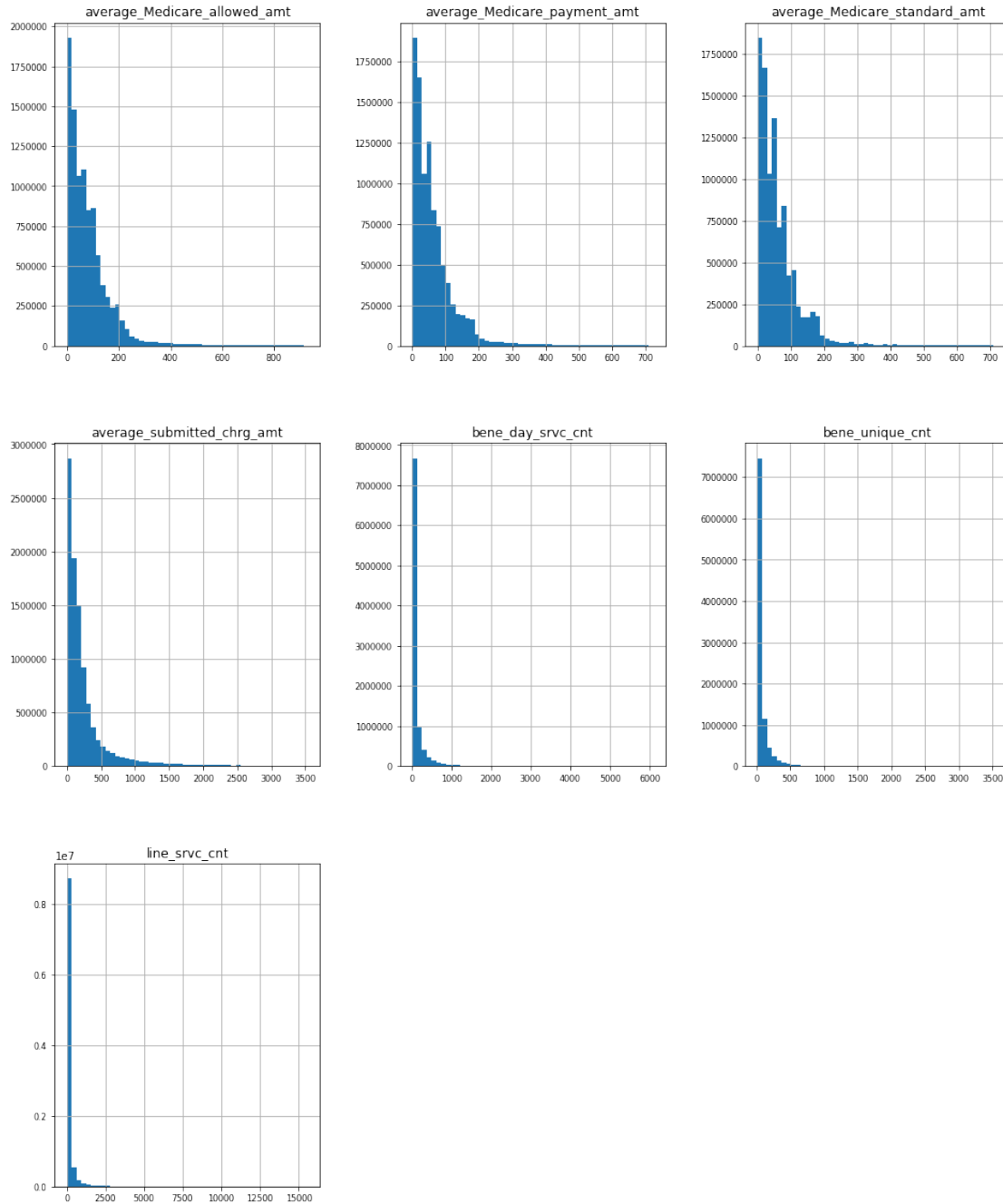
```
In [9]: fig, axes = plt.subplots(round(len(df_num.columns) / 3), 3, figsize=(12,9))

for i, ax in enumerate(fig.axes):
    if i < len(df_num.columns):
        sns.distplot(df_num.iloc[:,i], color='g', ax=ax) #hist_kws={'alpha': 0.4})

fig.tight_layout()
```



```
In [10]: df_num.hist(figsize=(16, 20), bins = 50, xlabelsize=8, ylabelsize=8);
```



```
In [11]: logx = np.log(df_num+1)
         logx.describe()
```

```
Out[11]:
```

	line_srv_cnt	bene_unique_cnt	bene_day_srv_cnt	\
count	9.689357e+06	9.689357e+06	9.689357e+06	
mean	4.049679e+00	3.719968e+00	3.969712e+00	
std	1.209221e+00	9.754828e-01	1.133209e+00	

min	2.140066e+00	2.484907e+00	2.484907e+00
25%	3.091042e+00	2.890372e+00	3.044522e+00
50%	3.784190e+00	3.526361e+00	3.737670e+00
75%	4.779123e+00	4.343805e+00	4.682131e+00
max	9.655859e+00	8.186186e+00	8.715552e+00

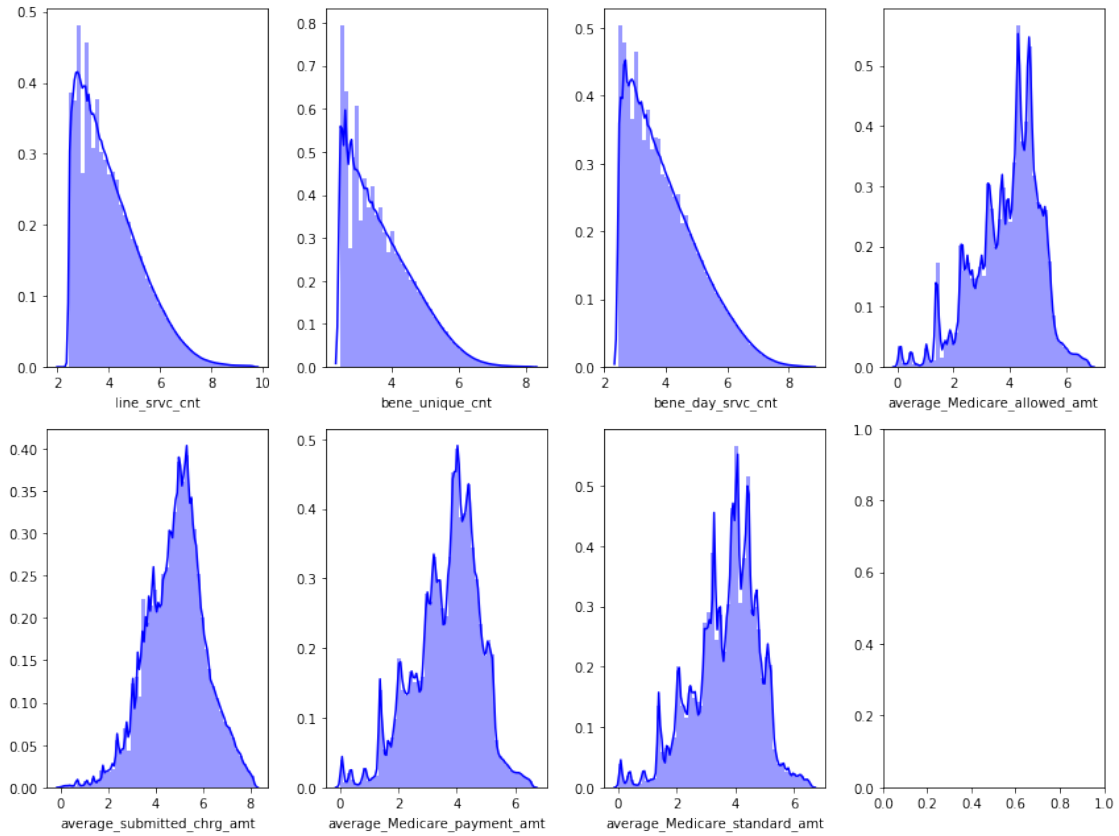
	average_Medicare_allowed_amt	average_submitted_chrg_amt \
count	9.689357e+06	9.689357e+06
mean	3.923733e+00	4.892136e+00
std	1.142726e+00	1.230041e+00
min	6.035198e-05	6.035198e-05
25%	3.223266e+00	4.060443e+00
50%	4.168833e+00	4.974644e+00
75%	4.722242e+00	5.666427e+00
max	6.837272e+00	8.173345e+00

	average_Medicare_payment_amt	average_Medicare_standard_amt
count	9.689357e+06	9.689357e+06
mean	3.667014e+00	3.681859e+00
std	1.103607e+00	1.099124e+00
min	0.000000e+00	0.000000e+00
25%	3.000201e+00	3.039022e+00
50%	3.857245e+00	3.875346e+00
75%	4.432918e+00	4.434959e+00
max	6.585347e+00	6.588716e+00

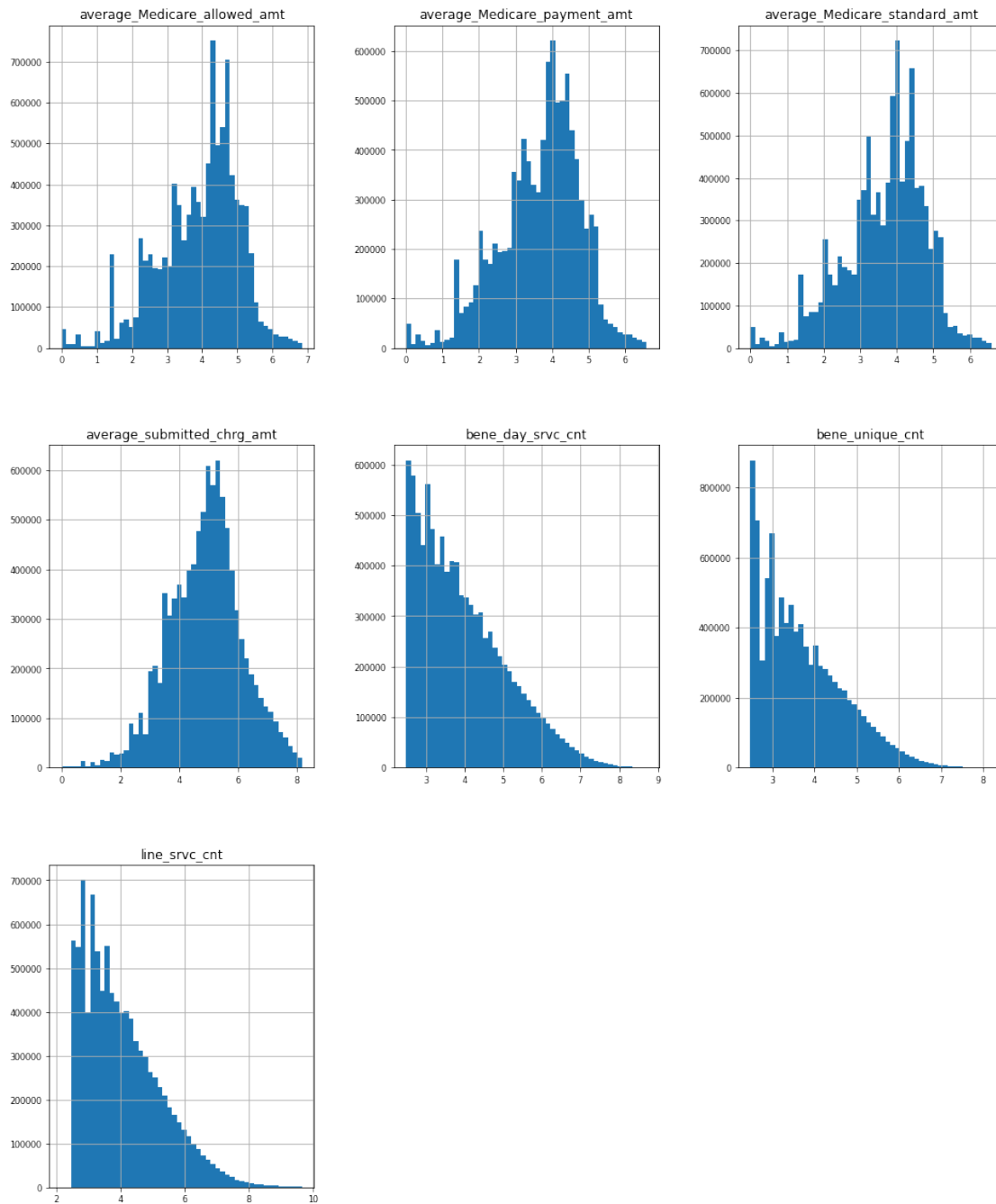
```
In [12]: fig, axes = plt.subplots(round(len(df_num.columns) / 3), 4, figsize=(12,9))
```

```
for i, ax in enumerate(fig.axes):
    if i < len(df_num.columns):
        sns.distplot(logx.iloc[:,i], color='b', ax=ax) #bins=100, hist_kws={'alpha': 0.5}

fig.tight_layout()
```



```
In [13]: logx.hist(figsize=(16, 20), bins = 50, xlabelsize=8, ylabelsize=8);
```



1.4 EDA of Non-Numeric Data

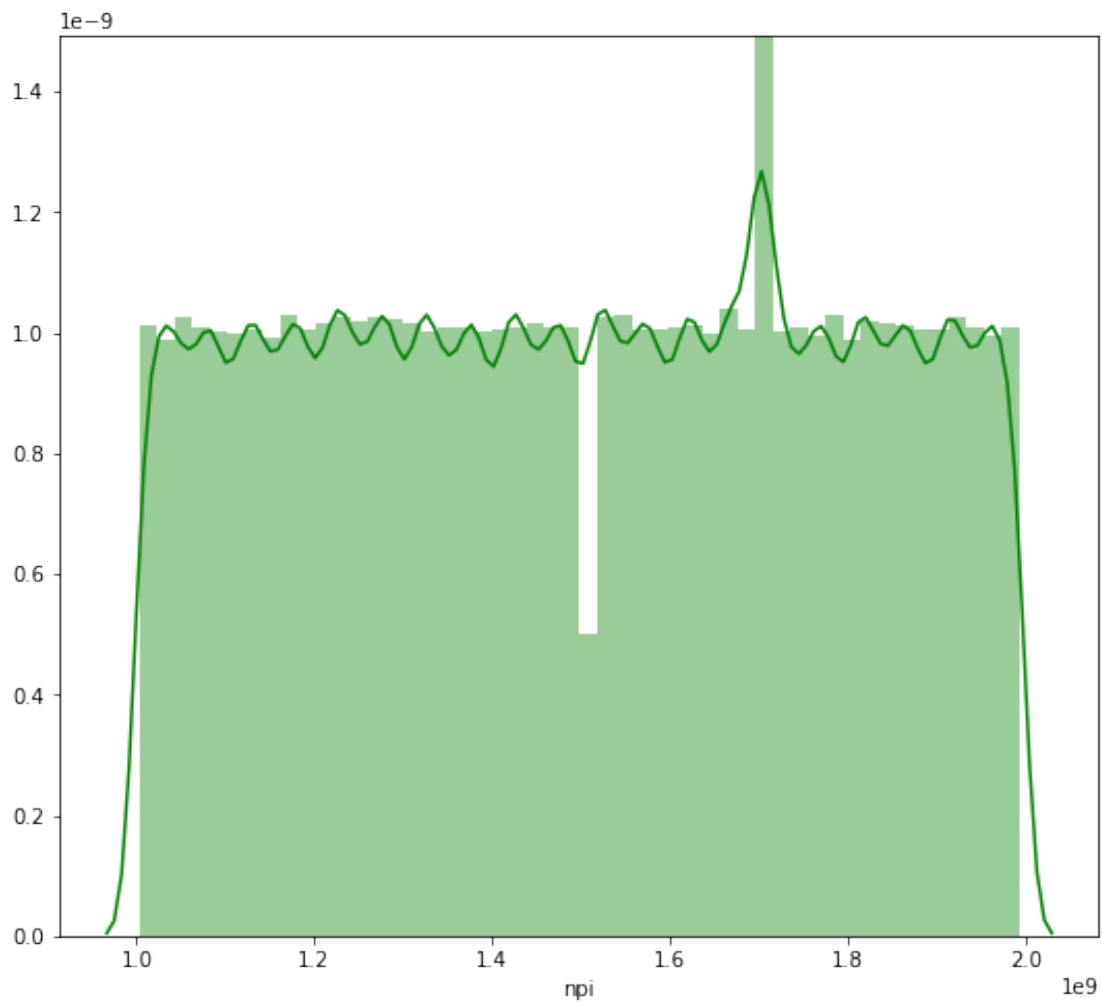
```
In [14]: print(df['npi'].describe())
plt.figure(figsize=(9, 8))
sns.distplot(df['npi'], color='g', hist_kws={'alpha': 0.4});
```

```
count    9.689357e+06
mean     1.499871e+09
```

```

std      2.877043e+08
min      1.003000e+09
25%      1.255302e+09
50%      1.497987e+09
75%      1.740680e+09
max      1.993000e+09
Name: np_i, dtype: float64

```



```

In [204]: print(df['nppes_credentials'].describe())
           #print(set(df['nppes_credentials']))

           # This output is extremely large, so it is left commented

count      8997201
unique      15968

```

```
top          MD
freq         3184587
Name: npes_credentials, dtype: object
```

```
In [205]: print(df['npes_provider_state'].describe())
          print(set(df['npes_provider_state']))
```

```
count      9689357
unique       61
top         CA
freq       745862
Name: npes_provider_state, dtype: object
{'OR', 'AP', 'WA', 'XX', 'HI', 'WV', 'WI', 'OH', 'MN', 'ND', 'IA', 'NE', 'UT', 'CO', 'TX', 'CT'}
```

```
In [17]: print(df['npes_provider_country'].describe())
          print(set(df['npes_provider_country']))
```

```
count      9689357
unique       25
top         US
freq      9688765
Name: npes_provider_country, dtype: object
{'MX', 'TH', 'TR', 'NO', 'TT', 'CR', 'SA', 'JP', 'US', 'CA', 'LB', 'AG', 'IN', 'CN', 'PK', 'DE'}
```

```
In [206]: print(df['hcpcs_code'].describe())
          #print(set(df['hcpcs_code']))
```

```
# With 5652 unique values, this output is also very long so it is commented out to s
```

```
count      9689357
unique      5652
top        99213
freq      457817
Name: hcpcs_code, dtype: object
```

```
In [19]: print(df['provider_type'].describe())
          print(set(df['provider_type']))
```

```
count      9689357
unique       94
top      Diagnostic Radiology
freq      1235785
Name: provider_type, dtype: object
{'Vascular Surgery', 'Pharmacy', 'Pathology', 'Neurosurgery', 'Plastic and Reconstructive Surg
```

1.5 EDA of Categorical Data

Selecting which categorical factors are appropriate to use in clustering.

```
In [20]: df_not_num = df.select_dtypes(include = ['O'])
        print('There is {} non numerical features including:\n{}'.format(len(df_not_num.columns), df_not_num.columns))
```

There is 18 non numerical features including:

['nppes_provider_last_org_name', 'nppes_provider_first_name', 'nppes_provider_mi', 'nppes_cred']

```
In [21]: df_imp_cat = df[['nppes_provider_gender', 'nppes_entity_code', 'nppes_provider_state', 'nppes_provider_country', 'provider_type', 'medicare_participation_indicator', 'place_of_service', 'hcpcs_drug_indicator']]
        print('There is {} non numerical features which are important or not too large in size'.format(len(df_imp_cat.columns)))
        df_imp_cat.info()
        df_imp_cat.describe()
```

There is 18 non numerical features which are important or not too large in size including:

['nppes_provider_last_org_name', 'nppes_provider_first_name', 'nppes_provider_mi', 'nppes_cred']

<class 'pandas.core.frame.DataFrame'>

Int64Index: 9689357 entries, 1 to 9847443

Data columns (total 8 columns):

nppes_provider_gender	object
nppes_entity_code	object
nppes_provider_state	object
nppes_provider_country	object
provider_type	object
medicare_participation_indicator	object
place_of_service	object
hcpcs_drug_indicator	object
dtypes:	object(8)
memory usage:	665.3+ MB

```
Out[21]:
```

	nppes_provider_gender	nppes_entity_code	nppes_provider_state	\
count	9295989	9689357	9689357	
unique	2	2	61	
top	M	I	CA	
freq	6429582	9295989	745862	

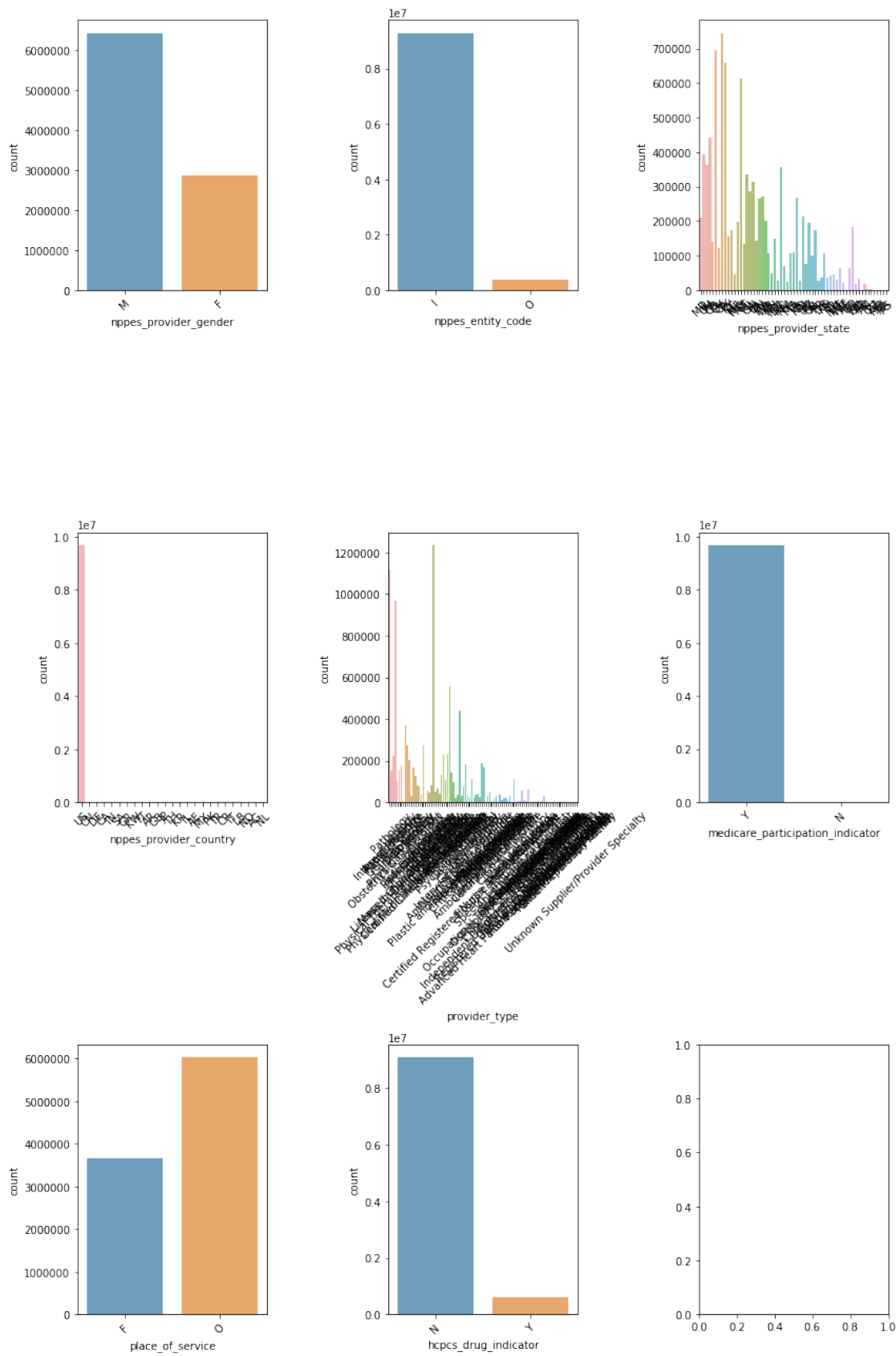
	nppes_provider_country	provider_type	\
count	9689357	9689357	
unique	25	94	
top	US	Diagnostic Radiology	
freq	9688765	1235785	

	medicare_participation_indicator	place_of_service	hcpcs_drug_indicator
count	9689357	9689357	9689357
unique	2	2	2
top	Y	0	N
freq	9686063	6029489	9088521

```
In [22]: fig, axes = plt.subplots(round(len(df_imp_cat.columns) / 3), 3, figsize=(12,18))

for i, ax in enumerate(fig.axes):
    if i < len(df_imp_cat.columns):
        ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)
        sns.countplot(x=df_imp_cat.columns[i], alpha=0.7, data=df_imp_cat, ax=ax)

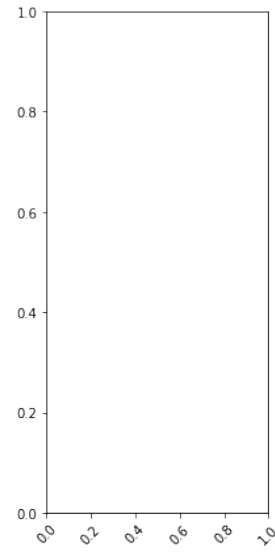
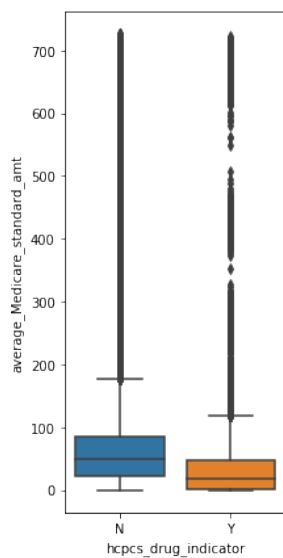
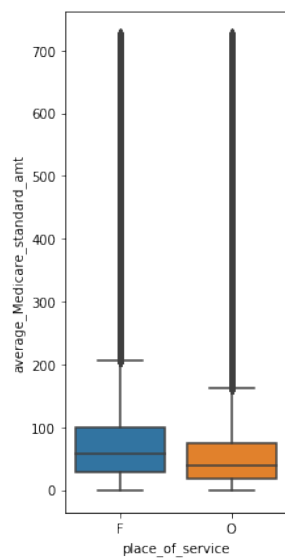
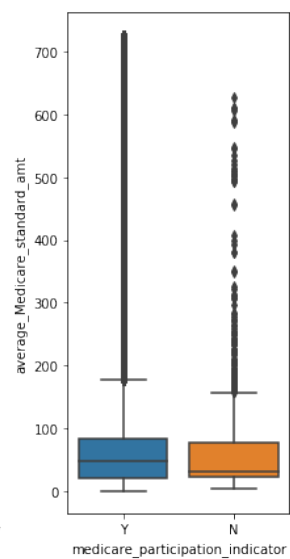
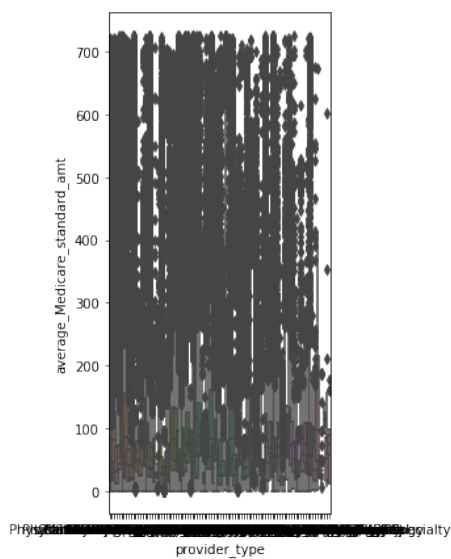
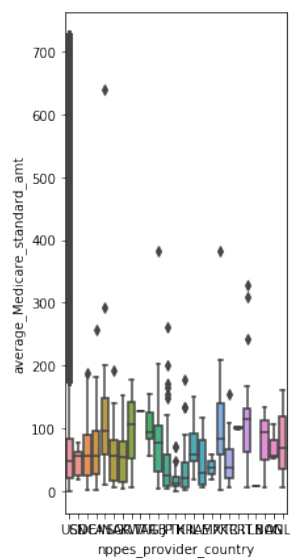
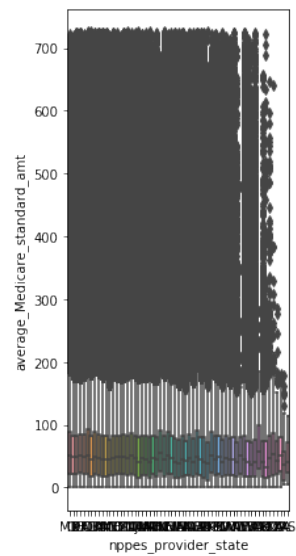
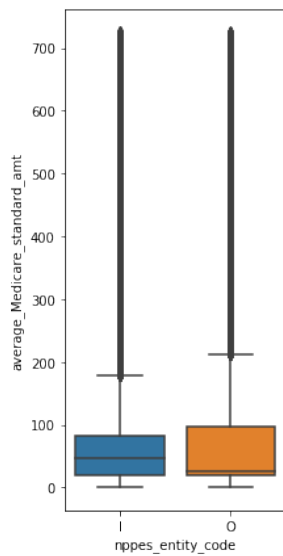
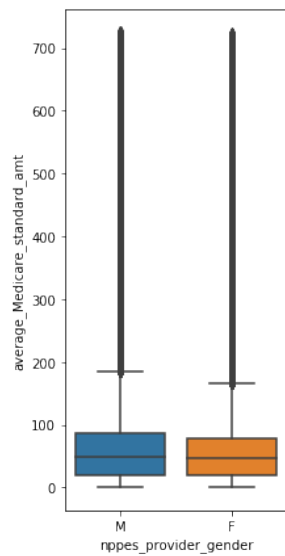
fig.tight_layout()
```




```
In [23]: fig, axes = plt.subplots(round(len(df_imp_cat.columns) / 3), 3, figsize=(12,18))

for i, ax in enumerate(fig.axes):
    if i < len(df_imp_cat.columns):
        sns.boxplot(x=df_imp_cat.columns[i], y='average_Medicare_standard_amt', data=
            #plt.setp(ax.artists, alpha=.5, linewidth=2, edgecolor="k")
        plt.xticks(rotation=45)

fig.tight_layout()
```



```
In [24]: fig, axes = plt.subplots(round(len(df_imp_cat.columns) / 3), 3, figsize=(12,18))

for i, ax in enumerate(fig.axes):
    if i < len(df_imp_cat.columns):
        sns.boxplot(x=df_imp_cat.columns[i], y='average_submitted_chrg_amt', data=df,
                    #plt.setp(ax.artists, alpha=.5, linewidth=2, edgecolor="k")
                    plt.xticks(rotation=45)

fig.tight_layout()
```



```

In [25]: # Creating a dummy variable for gender
pd.get_dummies(df['nppes_provider_gender'],prefix='gender')
df = pd.concat([df,pd.get_dummies(df['nppes_provider_gender'],prefix='gender')],axis=

In [26]: # Creating a dummy variable for the entity code
pd.get_dummies(df['nppes_entity_code'],prefix='entity_code')
df = pd.concat([df,pd.get_dummies(df['nppes_entity_code'],prefix='entity_code')],axis=

In [27]: # Creating a dummy variable for the entity code
pd.get_dummies(df['hcpcs_drug_indicator'],prefix='drug_indicator')
df = pd.concat([df,pd.get_dummies(df['hcpcs_drug_indicator'],prefix='drug_indicator')],axis=

In [28]: # Creating a dummy variable for the participation indicator
pd.get_dummies(df['medicare_participation_indicator'],prefix='participation_indicator')
df = pd.concat([df,pd.get_dummies(df['medicare_participation_indicator'],prefix='participation_indicator')],axis=

In [29]: # Creating a dummy variable for the participation indicator
pd.get_dummies(df['place_of_service'],prefix='place_of_service')
df = pd.concat([df,pd.get_dummies(df['place_of_service'],prefix='place_of_service')],axis=

In [30]: df.shape

Out[30]: (9689357, 36)

In [31]: df.head(5)

Out[31]:
```

	npi	nppes_provider_last_org_name	nppes_provider_first_name	\
1	1003000126	ENKESHAFI	ARDALAN	
2	1003000126	ENKESHAFI	ARDALAN	
3	1003000126	ENKESHAFI	ARDALAN	
4	1003000126	ENKESHAFI	ARDALAN	
5	1003000126	ENKESHAFI	ARDALAN	

	nppes_provider_mi	nppes_credentials	nppes_provider_gender	nppes_entity_code	\
1	NaN	M.D.	M	I	
2	NaN	M.D.	M	I	
3	NaN	M.D.	M	I	
4	NaN	M.D.	M	I	
5	NaN	M.D.	M	I	

	nppes_provider_street1	nppes_provider_street2	nppes_provider_city	...	\
1	900 SETON DR	NaN	CUMBERLAND	...	
2	900 SETON DR	NaN	CUMBERLAND	...	
3	900 SETON DR	NaN	CUMBERLAND	...	
4	900 SETON DR	NaN	CUMBERLAND	...	
5	900 SETON DR	NaN	CUMBERLAND	...	

	gender_F	gender_M	entity_code_I	entity_code_O	drug_indicator_N	\
1	0	1	1	0	1	
2	0	1	1	0	1	
3	0	1	1	0	1	
4	0	1	1	0	1	
5	0	1	1	0	1	

	drug_indicator_Y	participation_indicator_N	participation_indicator_Y	\
1	0	0	1	
2	0	0	1	
3	0	0	1	
4	0	0	1	
5	0	0	1	

	place_of_service_F	place_of_service_O
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0

[5 rows x 36 columns]

1.6 Start Clustering

Group by hcpcs codes 999201 - 99215, which are associated with new and returning outpatient visits. These codes are significant because they represent the most commonly used hcpcs codes. Then, I will cluster on gender, entity code, participation indicator, and drug indicator.

```
In [111]: df_clust = []
df_clust = df.loc[df['hcpcs_code'] > '99200']
df_clust = df_clust.loc[df_clust['hcpcs_code'] < '99215']
df_clust = df_clust.reset_index(drop=True)
df_clust.head()
```

```
Out[111]:      npi nnpes_provider_last_org_name nnpes_provider_first_name \
0  1003000142      KHALIL      RASHID
1  1003000142      KHALIL      RASHID
2  1003000142      KHALIL      RASHID
3  1003000142      KHALIL      RASHID
4  1003000142      KHALIL      RASHID
```

	nnpes_provider_mi	nnpes_credentials	nnpes_provider_gender	nnpes_entity_code	\
0	NaN	M.D.	M	I	
1	NaN	M.D.	M	I	
2	NaN	M.D.	M	I	
3	NaN	M.D.	M	I	

	4	NaN	M.D.	M	I
		nppes_provider_street1	nppes_provider_street2	nppes_provider_city	...
0	4126	N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...
1	4126	N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...
2	4126	N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...
3	4126	N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...
4	4126	N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...

	gender_F	gender_M	entity_code_I	entity_code_O	drug_indicator_N	\
0	0	1	1	0	1	
1	0	1	1	0	1	
2	0	1	1	0	1	
3	0	1	1	0	1	
4	0	1	1	0	1	

	drug_indicator_Y	participation_indicator_N	participation_indicator_Y	\
0	0	0	1	
1	0	0	1	
2	0	0	1	
3	0	0	1	
4	0	0	1	

	place_of_service_F	place_of_service_O
0	0	1
1	0	1
2	0	1
3	1	0
4	0	1

[5 rows x 36 columns]

```
In [116]: pay_ratio = logx['average_Medicare_allowed_amt']/logx['average_submitted_chrg_amt']
pay_ratio = pay_ratio.reset_index(drop=True)
pay_ratio.head()
```

```
Out[116]: 0    0.744382
1    0.755562
2    0.766590
3    0.792572
4    0.754399
dtype: float64
```

```
In [117]: df_clust['pay_ratio'] = pay_ratio
df_clust.head()
```

```
Out[117]:      npi nppes_provider_last_org_name nppes_provider_first_name \
0  1003000142                KHALIL                RASHID
1  1003000142                KHALIL                RASHID
```

2	1003000142	KHALIL	RASHID
3	1003000142	KHALIL	RASHID
4	1003000142	KHALIL	RASHID

	nppes_provider_mi	nppes_credentials	nppes_provider_gender	nppes_entity_code	\
0	NaN	M.D.	M	I	
1	NaN	M.D.	M	I	
2	NaN	M.D.	M	I	
3	NaN	M.D.	M	I	
4	NaN	M.D.	M	I	

	nppes_provider_street1	nppes_provider_street2	nppes_provider_city	...	\
0	4126 N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...	
1	4126 N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...	
2	4126 N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...	
3	4126 N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...	
4	4126 N HOLLAND SYLVANIA RD	SUITE 220	TOLEDO	...	

	gender_M	entity_code_I	entity_code_O	drug_indicator_N	drug_indicator_Y	\
0	1	1	0	1	0	
1	1	1	0	1	0	
2	1	1	0	1	0	
3	1	1	0	1	0	
4	1	1	0	1	0	

	participation_indicator_N	participation_indicator_Y	place_of_service_F	\
0	0	1	0	
1	0	1	0	
2	0	1	0	
3	0	1	1	
4	0	1	0	

	place_of_service_O	pay_ratio
0	1	0.744382
1	1	0.755562
2	1	0.766590
3	0	0.792572
4	1	0.754399

[5 rows x 37 columns]

```
In [118]: train_ind = df_clust.sample(frac = 0.9, random_state = 0).index
train = df_clust.loc[train_ind,:]
test = df_clust[~df_clust.index.isin(train_ind)]
```

```
In [119]: train.describe()
```

```
Out[119]:
```

	npi	line_srvc_cnt	bene_unique_cnt	bene_day_srvc_cnt	\
count	1.370557e+06	1.370557e+06	1.370557e+06	1.370557e+06	

mean	1.499709e+09	1.554753e+02	9.443000e+01	1.554671e+02
std	2.877891e+08	2.629306e+02	1.249527e+02	2.629181e+02
min	1.003000e+09	7.500000e+00	1.100000e+01	1.100000e+01
25%	1.245678e+09	2.500000e+01	2.200000e+01	2.500000e+01
50%	1.497965e+09	6.000000e+01	4.900000e+01	6.000000e+01
75%	1.740721e+09	1.670000e+02	1.160000e+02	1.670000e+02
max	1.993000e+09	6.058000e+03	3.503000e+03	6.057000e+03

	average_Medicare_allowed_amt	average_submitted_chrg_amt \
count	1.370557e+06	1.370557e+06
mean	9.284059e+01	1.939008e+02
std	4.154348e+01	1.192317e+02
min	1.000000e-02	1.000000e-02
25%	6.770000e+01	1.153279e+02
50%	8.425000e+01	1.661438e+02
75%	1.097500e+02	2.400000e+02
max	2.726200e+02	3.519767e+03

	average_Medicare_payment_amt	average_Medicare_standard_amt \
count	1.370557e+06	1.370557e+06
mean	6.348889e+01	6.494087e+01
std	3.064066e+01	3.027190e+01
min	0.000000e+00	0.000000e+00
25%	4.327579e+01	4.594652e+01
50%	5.728633e+01	5.784425e+01
75%	7.666340e+01	7.816500e+01
max	2.614739e+02	1.805492e+02

	gender_F	gender_M	entity_code_I	entity_code_O \
count	1.370557e+06	1.370557e+06	1.370557e+06	1.370557e+06
mean	3.608387e-01	6.391460e-01	9.999847e-01	1.532224e-05
std	4.802440e-01	4.802484e-01	3.914334e-03	3.914334e-03
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	0.000000e+00	0.000000e+00	1.000000e+00	0.000000e+00
50%	0.000000e+00	1.000000e+00	1.000000e+00	0.000000e+00
75%	1.000000e+00	1.000000e+00	1.000000e+00	0.000000e+00
max	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00

	drug_indicator_N	drug_indicator_Y	participation_indicator_N \
count	1370557.0	1370557.0	1.370557e+06
mean	1.0	0.0	4.479930e-04
std	0.0	0.0	2.116111e-02
min	1.0	0.0	0.000000e+00
25%	1.0	0.0	0.000000e+00
50%	1.0	0.0	0.000000e+00
75%	1.0	0.0	0.000000e+00
max	1.0	0.0	1.000000e+00

	participation_indicator_Y	place_of_service_F	place_of_service_0 \
count	1.370557e+06	1.370557e+06	1.370557e+06
mean	9.995520e-01	1.259517e-01	8.740483e-01
std	2.116111e-02	3.317951e-01	3.317951e-01
min	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.000000e+00	0.000000e+00	1.000000e+00
50%	1.000000e+00	0.000000e+00	1.000000e+00
75%	1.000000e+00	0.000000e+00	1.000000e+00
max	1.000000e+00	1.000000e+00	1.000000e+00

	pay_ratio
count	1.370557e+06
mean	7.973866e-01
std	1.360340e-01
min	3.049488e-03
25%	7.288828e-01
50%	8.157591e-01
75%	8.869255e-01
max	1.000000e+00

```
In [120]: set(train['hcpcs_code'])
```

```
Out[120]: {'99201',
           '99202',
           '99203',
           '99204',
           '99205',
           '99211',
           '99212',
           '99213',
           '99214'}
```

```
In [121]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1370557 entries, 1401931 to 1440521
Data columns (total 37 columns):
npi                                1370557 non-null int64
nppes_provider_last_org_name      1370532 non-null object
nppes_provider_first_name        1370516 non-null object
nppes_provider_mi                 1018444 non-null object
nppes_credentials                 1327933 non-null object
nppes_provider_gender             1370536 non-null object
nppes_entity_code                 1370557 non-null object
nppes_provider_street1           1370554 non-null object
nppes_provider_street2           611032 non-null object
nppes_provider_city               1370557 non-null object
nppes_provider_zip               1370557 non-null object
nppes_provider_state             1370557 non-null object
```

```

nppes_provider_country      1370557 non-null object
provider_type               1370557 non-null object
medicare_participation_indicator 1370557 non-null object
place_of_service           1370557 non-null object
hcpcs_code                 1370557 non-null object
hcpcs_description          1370557 non-null object
hcpcs_drug_indicator       1370557 non-null object
line_srv_cnt               1370557 non-null float64
bene_unique_cnt            1370557 non-null float64
bene_day_srv_cnt           1370557 non-null float64
average_Medicare_allowed_amt 1370557 non-null float64
average_submitted_chrg_amt  1370557 non-null float64
average_Medicare_payment_amt 1370557 non-null float64
average_Medicare_standard_amt 1370557 non-null float64
gender_F                   1370557 non-null uint8
gender_M                   1370557 non-null uint8
entity_code_I              1370557 non-null uint8
entity_code_O              1370557 non-null uint8
drug_indicator_N           1370557 non-null uint8
drug_indicator_Y           1370557 non-null uint8
participation_indicator_N  1370557 non-null uint8
participation_indicator_Y  1370557 non-null uint8
place_of_service_F         1370557 non-null uint8
place_of_service_O         1370557 non-null uint8
pay_ratio                  1370557 non-null float64
dtypes: float64(8), int64(1), object(18), uint8(10)
memory usage: 305.9+ MB

```

1.6.1 Removing all factors except those which will be clustered on

```
In [122]: train = train[['bene_day_srv_cnt', 'pay_ratio']]
          train.head()
```

```
Out[122]:
```

	bene_day_srv_cnt	pay_ratio
1401931	58.0	0.890901
245413	290.0	0.889787
1226995	474.0	0.828879
126031	144.0	0.768402
312578	212.0	0.909610

```
In [123]: train.describe()
```

```
Out[123]:
```

	bene_day_srv_cnt	pay_ratio
count	1.370557e+06	1.370557e+06
mean	1.554671e+02	7.973866e-01
std	2.629181e+02	1.360340e-01
min	1.100000e+01	3.049488e-03
25%	2.500000e+01	7.288828e-01

50%	6.000000e+01	8.157591e-01
75%	1.670000e+02	8.869255e-01
max	6.057000e+03	1.000000e+00

```
In [124]: scaler = preprocessing.StandardScaler().fit(train)
dfNorm = scaler.transform(train)
```

```
In [125]: train.head()
```

```
Out[125]:
```

	bene_day_srvc_cnt	pay_ratio
1401931	58.0	0.890901
245413	290.0	0.889787
1226995	474.0	0.828879
126031	144.0	0.768402
312578	212.0	0.909610

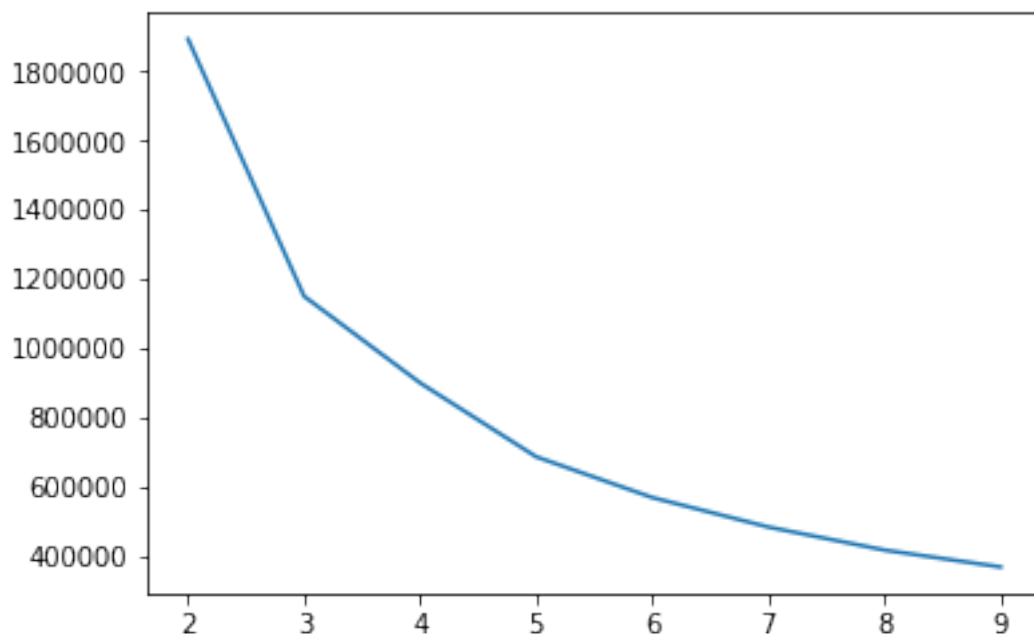
```
In [126]: maxClusters = 10
sse = []
for nClusters in range(2,maxClusters):
    kmeans = KMeans(n_clusters=nClusters, random_state=0).fit(dfNorm)
    sse.append(kmeans.inertia_)
```

```
In [127]: print(sse)
```

```
[1893657.8781051266, 1149673.0292939371, 898902.3803362981, 685103.9830502096, 567278.25903933]
```

```
In [128]: plt.plot(range(2,maxClusters),sse)
```

```
Out[128]: [<matplotlib.lines.Line2D at 0x1a4ea3fcf8>]
```



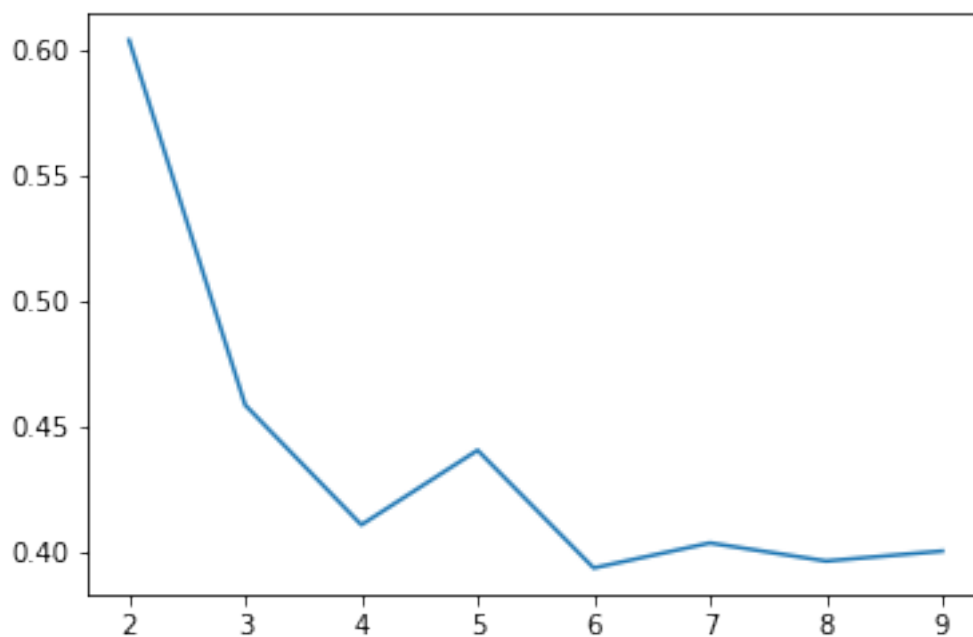
```
In [129]: silh = []
          for nClusters in range(2,maxClusters):
              kmeans = KMeans(n_clusters=nClusters, random_state=0).fit(dfNorm)
              silhouette_avg = silhouette_score(dfNorm, kmeans.labels_, sample_size = 10000)
              silh.append(silhouette_avg)
```

```
In [130]: print(silh)
```

```
[0.603865802545504, 0.4585196885202319, 0.41084431257905807, 0.440537862972748, 0.393665727786]
```

```
In [131]: plt.plot(range(2,maxClusters),silh)
```

```
Out[131]: [<matplotlib.lines.Line2D at 0x1a4e90c5c0>]
```



```
In [132]: range_n_clusters = [3, 5]
```

```
for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
```

```

# The silhouette coefficient can range from -1, 1 but in this example all
# lie within [-0.1, 1]
#ax1.set_xlim([-0.1, 1])
# The (n_clusters+1)*10 is for inserting blank space between silhouette
# plots of individual clusters, to demarcate them clearly.
#ax1.set_ylim([0, len(dfNorm) + (n_clusters + 1) * 10])

# Initialize the clusterer with n_clusters value and a random generator
# seed of 10 for reproducibility.
kmeans = KMeans(n_clusters=n_clusters, random_state=0)
cluster_labels = kmeans.fit_predict(dfNorm)

# The silhouette_score gives the average value for all the samples.
# This gives a perspective into the density and separation of the formed
# clusters
silhouette_avg = silhouette_score(dfNorm, kmeans.labels_, sample_size = 10000)
print("For n_clusters =", n_clusters,
      "The average silhouette_score is :", silhouette_avg)

indices = np.random.choice(dfNorm.shape[0], 10000, replace = False)

# Compute the silhouette scores for each sample
sample_silhouette_values = silhouette_samples(dfNorm[indices], kmeans.labels_[indices])

y_lower = 10
for i in range(n_clusters):
    # Aggregate the silhouette scores for samples belonging to
    # cluster i, and sort them
    ith_cluster_silhouette_values = sample_silhouette_values[kmeans.labels_[indices] == i]
    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cm.nipy_spectral(float(i) / n_clusters)

    ax1.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values, color)

    # Label the silhouette plots with their cluster numbers at the middle
    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    # Compute the new y_lower for next plot
    y_lower = y_upper + 10 # 10 for the 0 samples

ax1.set_title("The silhouette plot for the various clusters.")
ax1.set_xlabel("The silhouette coefficient values")
ax1.set_ylabel("Cluster label")

```

```

# The vertical line for average silhouette score of all the values
ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

ax1.set_yticks([]) # Clear the yaxis labels / ticks
ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

# 2nd Plot showing the actual clusters formed
colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
ax2.scatter(dfNorm[:, 0], dfNorm[:, 1], marker='.', s=30, lw=0, alpha=0.7, c=col

# Labeling the clusters
centers = kmeans.cluster_centers_
# Draw white circles at cluster centers
ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
            c="white", alpha=1, s=200, edgecolor='k')

for i, c in enumerate(centers):
    ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                s=50, edgecolor='k')

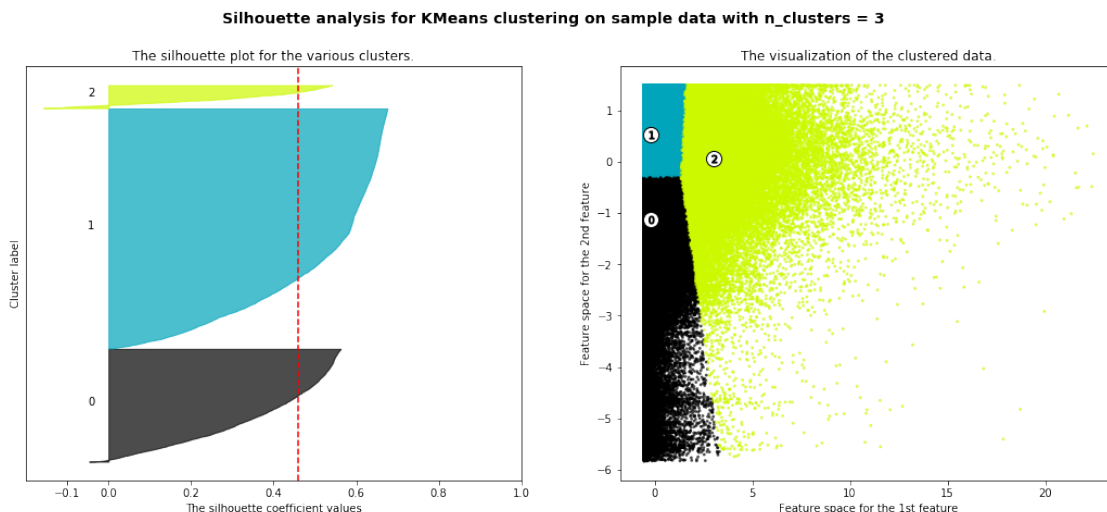
ax2.set_title("The visualization of the clustered data.")
ax2.set_xlabel("Feature space for the 1st feature")
ax2.set_ylabel("Feature space for the 2nd feature")

plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "
             "with n_clusters = %d" % n_clusters),
             fontsize=14, fontweight='bold')

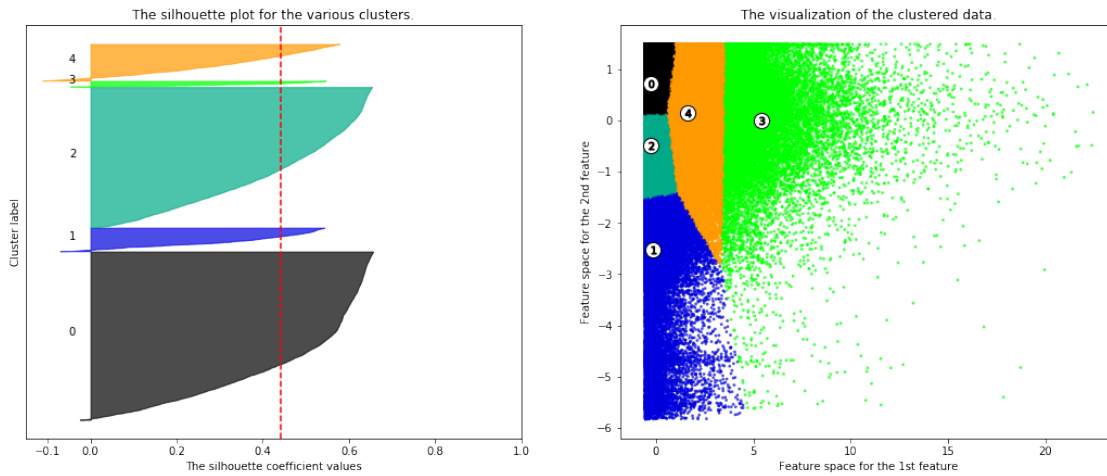
plt.show()

```

For n_clusters = 3 The average silhouette_score is : 0.46045892385587306
For n_clusters = 5 The average silhouette_score is : 0.44074056522550153



Silhouette analysis for KMeans clustering on sample data with n_clusters = 5



1.7 K-Means Clustering

```
In [133]: labels = KMeans(n_clusters=3, random_state=0).fit_predict(dfNorm)
```

```
In [134]: train['Clusters'] = labels
```

```
In [135]: train['Clusters'].value_counts()
```

```
Out[135]: 1      873195
          0      410814
          2      86548
          Name: Clusters, dtype: int64
```

```
In [136]: train.groupby('Clusters').mean()
```

```
Out[136]:
```

	bene_day_srvc_cnt	pay_ratio
Clusters		
0	105.597080	0.644874
1	100.448195	0.868207
2	947.276806	0.806798

Analyzing the clusters based on zip codes. The zip codes are broken down to their first digit to represent a specific region of the U.S.

```
In [187]: #df_a = []
          #df_a = df.loc[df['hcpcs_code'] > '99200']
          #df_a = df_a.loc[df_a['hcpcs_code'] < '99215']
          #print(df_a.iloc[:,10])
          #df_a.loc[0, 'nppes_provider_zip']
          df_a = df_clust.loc[train_ind,:]
          df_a.head()
```



```

Out[187]:
      npi nnpes_provider_last_org_name nnpes_provider_first_name \
1401931 1922011782          NASUR          ALI
245413  1164428801          CARTER        WILLIAM
1226995 1801854757          WILSON          JOHN
126031  1083648703          MALL          RONALD
312578  1205836533          KALLET        CYNTHIA

      nnpes_provider_mi nnpes_credentials nnpes_provider_gender \
1401931          M          M.D.          M
245413          H          MD          M
1226995          W          DO          M
126031          M          DO          M
312578          E          DO          F

      nnpes_entity_code      nnpes_provider_street1 \
1401931          I  3270 JOE BATTLE BLVD STE 380
245413          I          6201 CENTREVILLE RD
1226995          I          2011 N COLLINS BLVD
126031          I          3800 EAST BAY DRIVE
312578          I          3328 S SMITHVILLE RD

      nnpes_provider_street2 nnpes_provider_city ... gender_M \
1401931          NaN          EL PASO ...          1
245413          STE 100          CENTREVILLE ...          1
1226995          SUITE 609          RICHARDSON ...          1
126031  EAST BAY MEDICAL CENTER          LARGO ...          1
312578          NaN          DAYTON ...          0

      entity_code_I entity_code_O drug_indicator_N drug_indicator_Y \
1401931          1          0          1          0
245413          1          0          1          0
1226995          1          0          1          0
126031          1          0          1          0
312578          1          0          1          0

      participation_indicator_N participation_indicator_Y \
1401931          0          1
245413          0          1
1226995          0          1
126031          0          1
312578          0          1

      place_of_service_F place_of_service_O pay_ratio
1401931          0          1  0.890901
245413          0          1  0.889787
1226995          0          1  0.828879
126031          0          1  0.768402
312578          0          1  0.909610

```

[5 rows x 37 columns]

```
In [188]: def region(str):  
          str = str[0]  
          return(str)
```

```
In [189]: #train = train.reset_index(drop=True)  
          df_a['nppes_provider_zip'] = df_a['nppes_provider_zip'].astype(str)
```

```
In [190]: zip = []  
          for i in range(len(df_a)):  
              i = df_a.iloc[i,10]  
              zip.append(region(i))
```

```
In [191]: zip_array = pd.DataFrame(np.array([zip]).T)  
          zip_array.head()
```

```
Out[191]:    0  
0    7  
1    2  
2    7  
3    3  
4    4
```

```
In [192]: #zip_array = pd.DataFrame(np.array([zip]).T)  
          #zip_array = zip_array.reset_index(drop = True)  
          df_a = df_a.reset_index(drop = True)  
          df_a['zip'] = zip_array  
          #df = pd.concat([df,pd.get_dummies(df['place_of_service'],prefix='place_of_service')])  
          #set(df_a['zip'])  
          df_a.head()
```

```
Out[192]:      npi nppes_provider_last_org_name nppes_provider_first_name \  
0  1922011782                               NASUR                     ALI  
1  1164428801                               CARTER                   WILLIAM  
2  1801854757                               WILSON                     JOHN  
3  1083648703                               MALL                      RONALD  
4  1205836533                               KALLET                     CYNTHIA  
  
      nppes_provider_mi nppes_credentials nppes_provider_gender nppes_entity_code \  
0                      M                M.D.                   M                I  
1                      H                MD                     M                I  
2                      W                DO                     M                I  
3                      M                DO                     M                I  
4                      E                DO                     F                I  
  
      nppes_provider_street1 nppes_provider_street2 nppes_provider_city \  
0  3270 JOE BATTLE BLVD STE 380                NaN                EL PASO
```

1	6201 CENTREVILLE RD	STE 100	CENTREVILLE
2	2011 N COLLINS BLVD	SUITE 609	RICHARDSON
3	3800 EAST BAY DRIVE	EAST BAY MEDICAL CENTER	LARGO
4	3328 S SMITHVILLE RD	NaN	DAYTON

	... entity_code_I	entity_code_O	drug_indicator_N	drug_indicator_Y	\
0	...	1	0	1	0
1	...	1	0	1	0
2	...	1	0	1	0
3	...	1	0	1	0
4	...	1	0	1	0

	participation_indicator_N	participation_indicator_Y	place_of_service_F	\
0		0	1	0
1		0	1	0
2		0	1	0
3		0	1	0
4		0	1	0

	place_of_service_O	pay_ratio	zip
0	1	0.890901	7
1	1	0.889787	2
2	1	0.828879	7
3	1	0.768402	3
4	1	0.909610	4

[5 rows x 38 columns]

```
In [193]: train.head()
```

```
Out[193]:
```

	bene_day_srvc_cnt	pay_ratio	Clusters
0	58.0	0.890901	1
1	290.0	0.889787	1
2	474.0	0.828879	1
3	144.0	0.768402	1
4	212.0	0.909610	1

```
In [194]: finalDF = pd.concat([train,df_a['zip']],axis=1)
finalDF.head()
```

```
Out[194]:
```

	bene_day_srvc_cnt	pay_ratio	Clusters	zip
0	58.0	0.890901	1	7
1	290.0	0.889787	1	2
2	474.0	0.828879	1	7
3	144.0	0.768402	1	3
4	212.0	0.909610	1	4

```
In [195]: for i in range(3):
            print("Cluster:", i)
            print(finalDF.loc[finalDF.Clusters == i]['zip'].value_counts())
```

```

Cluster: 0
3    60383
1    53396
2    53051
4    49759
9    48902
7    46641
6    38003
8    31419
5    26298
0     2959
M         1
P         1
N         1
Name: zip, dtype: int64
Cluster: 1
3    128321
2    113310
1    113074
4    105110
9    104643
7     98528
6     81115
8     67047
5     55754
0      6280
M         4
N         4
L         2
A         2
B         1
Name: zip, dtype: int64
Cluster: 2
3    17137
2    12914
7    11853
9    10200
1     9961
4     8684
6     6990
8     6153
5     2148
0       508
Name: zip, dtype: int64

```

```

In [196]: finalDF = pd.concat([finalDF,pd.get_dummies(finalDF['zip'],prefix='zip')],axis=1)
          finalDF.head()

```

```

Out[196]:    bene_day_srvc_cnt  pay_ratio  Clusters  zip  zip_0  zip_1  zip_2  zip_3  \
0          58.0    0.890901         1    7      0      0      0      0
1         290.0    0.889787         1    2      0      0      1      0
2         474.0    0.828879         1    7      0      0      0      0
3         144.0    0.768402         1    3      0      0      0      1
4         212.0    0.909610         1    4      0      0      0      0

      zip_4  zip_5  zip_6  zip_7  zip_8  zip_9  zip_A  zip_B  zip_L  zip_M  \
0         0      0      0      1      0      0      0      0      0      0
1         0      0      0      0      0      0      0      0      0      0
2         0      0      0      1      0      0      0      0      0      0
3         0      0      0      0      0      0      0      0      0      0
4         1      0      0      0      0      0      0      0      0      0

      zip_N  zip_P
0         0      0
1         0      0
2         0      0
3         0      0
4         0      0

```

```

In [197]: finalDF.drop(['zip', 'zip_A', 'zip_B', 'zip_L', 'zip_M', 'zip_P', 'zip_N'], axis=1, inplace=True)
          finalDF.head()

```

```

Out[197]:    bene_day_srvc_cnt  pay_ratio  Clusters  zip_0  zip_1  zip_2  zip_3  zip_4  \
0          58.0    0.890901         1      0      0      0      0      0
1         290.0    0.889787         1      0      0      1      0      0
2         474.0    0.828879         1      0      0      0      0      0
3         144.0    0.768402         1      0      0      0      1      0
4         212.0    0.909610         1      0      0      0      0      1

      zip_5  zip_6  zip_7  zip_8  zip_9
0         0      0      1      0      0
1         0      0      0      0      0
2         0      0      1      0      0
3         0      0      0      0      0
4         0      0      0      0      0

```

```

In [198]: finalDF.groupby('Clusters').mean()

```

```

Out[198]:    bene_day_srvc_cnt  pay_ratio  zip_0  zip_1  zip_2  \
Clusters
0          105.597080    0.644874  0.007203  0.129976  0.129136
1          100.448195    0.868207  0.007192  0.129495  0.129765
2          947.276806    0.806798  0.005870  0.115092  0.149212

      zip_3  zip_4  zip_5  zip_6  zip_7  zip_8  zip_9
Clusters
0      0.146984  0.121123  0.064014  0.092507  0.113533  0.076480  0.119037

```

1	0.146956	0.120374	0.063851	0.092894	0.112836	0.076784	0.119839
2	0.198006	0.100337	0.024819	0.080764	0.136953	0.071093	0.117854

1.8 Testing

```
In [199]: test = test[['bene_day_srvc_cnt', 'pay_ratio']]
test.head()
```

```
Out[199]:
```

	bene_day_srvc_cnt	pay_ratio
4	330.0	0.754399
26	29.0	0.807190
38	214.0	0.928515
54	21.0	0.978910
75	45.0	0.798416

```
In [200]: scaler2 = preprocessing.StandardScaler().fit(test)
dfNorm2 = scaler.transform(test)
```

```
In [201]: labels2 = KMeans(n_clusters=3, random_state=0).fit_predict(dfNorm2)
test['Clusters'] = labels2
test['Clusters'].value_counts()
```

```
Out[201]:
```

2	97165
1	45202
0	9917

Name: Clusters, dtype: int64

```
In [202]: test.groupby('Clusters').mean()
```

```
Out[202]:
```

	bene_day_srvc_cnt	pay_ratio
Clusters		
0	937.753151	0.807853
1	106.421486	0.643872
2	99.052457	0.867735

```
In [ ]:
```