# Energy – Commercial Part

**NetId: jk2145**

## Part 0 – Data Set Description

The commercial part has eleven attributes: city, state_abbr, elec_score, gas_score, num_establishments, elec_1kdollars, elec_mwh, gas_1kdollars, gas_mcf, elec_lb_ghg, gas_lb_ghg:

- ✓ The combination of city and state_abbr is the key of one record.
- ✓ elec_score is used to show the weight of electricity use in every industry of one city, which related to the total electricity use in every industry in the United States.
- ✓ gas_score is used to show the weight of gas use in every industry of one city, which related to the total gas use in every industry in the United States.
- ✓ num_establishments represents the number of commercial buildings in one city.
- ✓ elec_1kdollars represents the electricity expenditure of one city.
- ✓ elec_mwh represents the electricity usage.
- ✓ gas_1kdollars represents the gas expenditure of one city.
- ✓ gas_mcf represents the gas usage.
- ✓ elec_lb_ghg represents how much greenhouse gas is produced per megawatt-hour electricity.
- ✓ gas_lb_ghg represents how much greenhouse gas is produced per thousand cubic feet gas.

## Part 1 – Basic Statistical Analysis and Data Cleaning Insight

### Part 1.1 – Re-processed Dataset

In order to compact the attribute relationship and facilitate subsequent data analysis and prediction work, this part combines two datasets in *"cleaned_commercial_building.csv*" in original "building and industrial" section and the dataset *"common_attrs_commercial.csv"* in original "electricity and natural gas" section. It drops some unrelated columns and duplicated rows, combines the rows based on the same city, and generates two new columns named elec_score and gas_score. Moreover, it generates the final dataset named *"energy_commercial.csv"*.

Firstly, this project analyzes the dataset and finds that the same city can have different zip codes. Since when it queries the data, it uses zip code as a request parameter and then may generate data for the same city multiple times. Furthermore, because different states may have the same city name, this project uses the combination of state and city name to identify one city uniquely. In this way, it drops those duplicate rows. Also, since column "index" and column "zip" are useless for the dataset, this project drops them. This process reduces the number of rows from 1433143 to 1142853.

Secondly, this project calculates two new attributes: elec_score and gas_score. Each row of the dataset "cleaned_commercial_building.csv" represents an industry of a city's energy usage data.

Since a city can have many industrials, there are many rows for one city in the dataset, which is hugely redundant. Therefore, this project calculates the weight of each industry named elec_score and gas_score and uses weight to replace the multiple rows. In this way, each city has only one row. The method to calculate weight is: 1. Calculate the average electricity usage of a building in one industry; 2. For each city, multiply the city's building number and corresponding average value. This process reduces the number of rows from 1142853 to 16821.

Thirdly, there are over 40 attributes in two datasets. This project drops some attributes which are not representative(e.g., for attribute 'elec_max_lb_ghg' and 'elec_lb_ghg', this project choose 'elec_lb_ghg' because it can reflect the average energy usage condition of a city), redundant(e.g., all rows have same value in attribute type), and irrelevant(e.g., attribute 'rank_of_electricity_use_per_establishment' is the rank not in the United State but in an industry).

Fourthly, this project merges these two datasets. Since few cities do not appear in both datasets, the merge result has missing value. After dropping the missing value, the amount of dataset is 16818.

At last, this project keeps these attributes: city, state_abbr, elec_score, gas_score, num_establishments, elec_1kdollars, elec_mwh, gas_1kdollars, gas_mcf, elec_lb_ghg, gas_lb_ghg.
Note: the missing value in the original dataset has been replaced by the nearest value or dropped.

**Part 1.2 – Mean, Median, and Standard Deviation**
In this part, I generate a file named *"statistic.csv"* to store the result. And the file's content is shown below:

| attribute | mean | median | std |
|---|---|---|---|
| elec_score | 281546.3 | 6221.92 | 10606141 |
| gas_score | 496516.42 | 10168.81 | 17491693.2 |
| num_establishments | 213.5 | 42 | 1382.86 |
| elec_1kdollars | 5028.7 | 417 | 53992.7 |
| elec_mwh | 43758.6 | 4199 | 367330.08 |
| gas_1kdollars | 921.3 | 47 | 8260.49 |
| gas_mcf | 127979.88 | 7021 | 1270724.03 |
| elec_lb_ghg | 65512410.5 | 6139039 | 553659068 |
| gas_lb_ghg | 15363048.4 | 842786 | 152541133 |

From the table, I find that each attribute's difference between mean and median are huge, which indicates that these attributes may have extreme value(outliers) to affect the results. The substantial standard deviation of each attribute can also prove this.

**Part 1.3 – Detect Outliers**

Since I realized that this dataset had outliers, I re-calculate the statistics of this dataset and add more metrics: min, 25%, 50%, 75%, max, range, var(means variance), dis(interquartile range). The result is stored in a file named *"statistic_original.csv"*. I find that each attribute has vast number: take num_establishment as an example, this attribute's median is 42, while its mean is about 5 times of its median, and its minimum value is 0 and the first 75% value of it is 145 while its maximum value is 92276, which is a very large number.

| | elec_score | gas_score | num_establi: | elec_1kdolla | elec_mwh | gas_1kdollar | gas_mcf | elec_lb_ghg | gas_lb_ghg |
|---|---|---|---|---|---|---|---|---|---|
| count | 16817 | 16817 | 16817 | 16817 | 16817 | 16817 | 16817 | 16817 | 16817 |
| mean | 281546.3 | 496516.42 | 213.5 | 5028.7 | 43758.6 | 921.3 | 127979.88 | 65512410.5 | 15363048.4 |
| std | 10606141 | 17491693.2 | 1382.86 | 53992.7 | 367330.08 | 8260.49 | 1270724.03 | 553659067.9 | 152541132.8 |
| min | 26.37 | 54.15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25% | 1101.23 | 1705.79 | 12 | 111 | 1148 | 4 | 549 | 1588394 | 65919 |
| 50% | 6221.92 | 10168.81 | 42 | 417 | 4199 | 47 | 7021 | 6139039 | 842786 |
| 75% | 36577.42 | 69149.05 | 145 | 2091 | 20216 | 346 | 49468 | 30225256 | 5938224 |
| max | 1184669438 | 1913917589 | 92276 | 4291287 | 26024809 | 564887 | 97518263 | 38995524672 | 11706354721 |
| median | 6221.92 | 10168.81 | 42 | 417 | 4199 | 47 | 7021 | 6139039 | 842786 |
| range | 1184669411 | 1913917535 | 92276 | 4291287 | 26024809 | 564887 | 97518263 | 38995524672 | 11706354721 |
| var | 37.67 | 35.23 | 6.48 | 10.74 | 8.39 | 8.97 | 9.93 | 8.45 | 9.93 |
| dis | 35476.19 | 67443.27 | 133 | 1980 | 19068 | 342 | 48919 | 28636862 | 5872305 |

Then, I introduce boxplot to handle these maximum values. In boxplot, the number which greater than 1.5IQR times of the upper quartile or less than 1.5IQR times of the lower quartile will be identified as outliers. And I calculate the number of outliers in each attribute. Here is the result:

```
elec_score  has  2677  outliers
gas_score  has  2608  outliers
num_establishments  has  2026  outliers
elec_1kdollars  has  2462  outliers
elec_mwh  has  2390  outliers
gas_1kdollars  has  2532  outliers
gas_mcf  has  2493  outliers
elec_lb_ghg  has  2394  outliers
gas_lb_ghg  has  2493  outliers
```

Then I replace them with the attribute corresponding median(calculated in part 1.2). And I re-calculate the statistics of this dataset and get the result, which is stored in file named *"statistic_revised.csv"*:

| | elec_score | gas_score | num_establi: | elec_1kdolla | elec_mwh | gas_1kdollar | gas_mcf | elec_lb_ghg | gas_lb_ghg |
|---|---|---|---|---|---|---|---|---|---|
| count | 16817 | 16817 | 16817 | 16817 | 16817 | 16817 | 16817 | 16817 | 16817 |
| mean | 11425.96 | 21320.07 | 62.16 | 726.95 | 7258.95 | 105.18 | 15277.67 | 10665914.1 | 1833967.11 |
| std | 17145.46 | 33201.96 | 73.29 | 1022.81 | 10058.4 | 174.44 | 24906.57 | 14926631.7 | 2989853.66 |
| min | 26.37 | 54.15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25% | 1101.23 | 1705.79 | 12 | 111 | 1148 | 4 | 549 | 1588394 | 65919 |
| 50% | 6221.92 | 10168.81 | 42 | 417 | 4199 | 47 | 7021 | 6139039 | 842786 |
| 75% | 12461.42 | 21965.82 | 77 | 770 | 7879 | 103 | 15379 | 11488218 | 1846141 |
| max | 89760.26 | 170258.52 | 344 | 5058 | 48783 | 859 | 122769 | 73138093 | 14737549 |
| median | 6221.92 | 10168.81 | 42 | 417 | 4199 | 47 | 7021 | 6139039 | 842786 |
| range | 89733.89 | 170204.37 | 344 | 5058 | 48783 | 859 | 122769 | 73138093 | 14737549 |
| var | 1.5 | 1.56 | 1.18 | 1.41 | 1.39 | 1.66 | 1.63 | 1.4 | 1.63 |
| dis | 11360.19 | 20260.03 | 65 | 659 | 6731 | 99 | 14830 | 9899824 | 1780222 |

The value of statistics' mean, std, max, range, var, dis changed significantly. Take num_establishment as an example, its std value changed from 1382.86 to 73.29, its maximum value changed from 92276 to 344, and its variance value changed from 6.48 to 1.18. In this way, I handle the maximum value in this dataset and store new dataset into the file named "*cleaned_energy_commercial.csv*".

**Part 1.4 – Local Outlier Factors**

In this part, I use LOF to handle high dimensional outliers. I assign value to k of 50, 100, and 200. First of all, I set the contamination to 0.1 and get outlier array and its score(generated by negative_outlier_factor_). I observe the result and find that some score is -3107757274153480, and it should be deleted while most of the outliers' score is about 1.2. I think it's not reasonable to delete them all. Therefore I only drop the score, which is lower than -2. I calculate the number of outliers for each k and the result is shown below:

```
For k= 50 : the outlier number is  340
For k= 100 : the outlier number is  422
For k= 200 : the outlier number is  580
```

I also sort the dataset based on the score and generate a file named "*lof_score.csv*" to store the result. The format is like:

| city | state_abbr | elec_score | gas_score | num_establi | elec_1kdolla | elec_mwh | gas_1kdollar | gas_mcf | elec_lb_ghg | gas_lb_ghg | lof_outlier | lof_score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lawrence | NY | 6125.07017 | 10265.1882 | 48 | 345 | 3389 | 81 | 10763 | 6392642 | 1292026 | -1 | -3.108E+15 |
| Rector | AR | 4394.04895 | 7708.63399 | 30 | 330 | 4008 | 76 | 10706 | 6395605 | 1285224 | -1 | -2.956E+15 |
| Sharon Sprin | KS | 4891.69696 | 7971.2755 | 30 | 456 | 3579 | 70 | 10360 | 6444729 | 1243614 | -1 | -2.869E+15 |
| Far Hills | NJ | 5893.2934 | 9736.57393 | 25 | 364 | 3114 | 81 | 11242 | 6000280 | 1349501 | -1 | -2.801E+15 |
| Lebanon | NH | 6221.92 | 10168.81 | 42 | 417 | 4199 | 107 | 10078 | 6139039 | 1209817 | -1 | -2.78E+15 |
| Kamiah | ID | 4667.66692 | 8866.64593 | 23 | 450 | 5056 | 72 | 9843 | 6236051 | 1181575 | -1 | -2.713E+15 |
| Wayne | NY | 5784.90242 | 11434.0185 | 54 | 314 | 3084 | 71 | 10121 | 5856044 | 1215019 | -1 | -2.712E+15 |
| Nederland | CO | 5903.87604 | 10180.1995 | 48 | 309 | 3338 | 51 | 9384 | 5848726 | 1126435 | -1 | -2.683E+15 |

I drop the outliers when k = 200 and generate a new dataset into file named "*lof_energy_commercial.csv*".

**Part 1.5 – Bin Strategy**

In this part, I choose two numeric attributes to bin. There are "elec_mwh" and "gas_mcf". Because I want to use this dataset to predict energy usage and elec_mwh represents the electricity usage, and "gas_mcf" represents the gas usage, I think it's acceptable to classify them into different labels and then it can be easy to do prediction work. I use equal-width bin method to bin these two attributes since I want to set the label based on the energy usage instead of the city numbers. Since I observe the dataset and find that there are many cities in which energy usage are low while only a few of them have high energy usage. Therefore, it is unreasonable to divide cities evenly according to their number. The bin result is shown below:

```
For elec_mwh , the bin result is:
(-1.0, 9756.0]        12698
(9756.0, 19513.0]      1725
(19513.0, 29270.0]      828
(29270.0, 39027.0]      600
(39027.0, 48784.0]      386
Name: elec_mwh_bin, dtype: int64
For gas_mcf , the bin result is:
(-1.0, 24553.2]        13093
(24553.2, 49107.4]      1421
(49107.4, 73661.6]       776
(73661.6, 98215.8]       539
(98215.8, 122770.0]      408
Name: gas_mcf_bin, dtype: int64
```

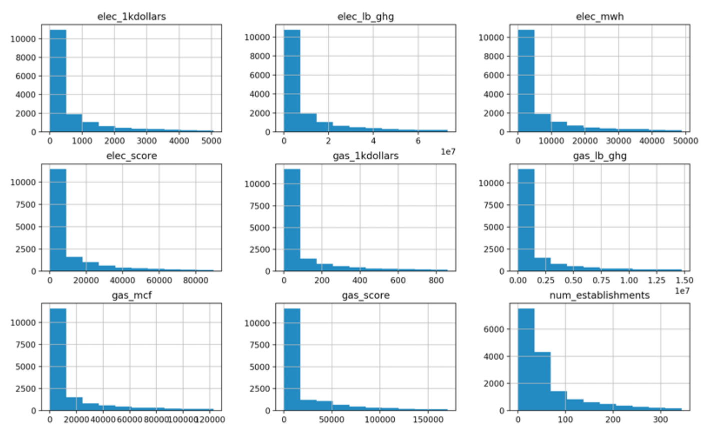
From this result, it is easy to observe that both the electricity usage and gas usage are concentrated in the first box. Then I divide the city into three levels: the first box corresponds to label "*Normal*", the second and third box correspond to label "*High*", and the fourth and fifth box correspond to label "*Extremely High*". I generate two new columns named "*elec_class*" and "*gas_class*" to store the labels. And I also generate a new file named "*labled_energy_commercial.csv*" to store the new dataset. The format of this new dataset is like:

| city | state_abbr | elec_score | gas_score | num_establi | elec_1kdolla | elec_mwh | gas_1kdollar | gas_mcf | elec_lb_ghg | gas_lb_ghg | elec_class | gas_class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Abbeville | LA | 22738.7384 | 36118.0366 | 315 | 3668 | 35874 | 585 | 73920 | 57242436 | 8873594 | Extremely High | Extremely High |
| Abbot | ME | 2288.7998 | 4329.58553 | 5 | 23 | 190 | 0 | 0 | 206707 | 0 | Normal | Normal |
| Abbotsford | WI | 18603.703 | 26444.4053 | 61 | 592 | 5968 | 108 | 16440 | 6460008 | 1973481 | Normal | Normal |
| Abbott | PA | 297.68077 | 651.638925 | 6 | 25 | 274 | 5 | 683 | 527464 | 81991 | Normal | Normal |
| Abbott | TX | 531.182058 | 928.267336 | 1 | 7 | 85 | 1 | 116 | 113766 | 13929 | Normal | Normal |
| Abbottstown | PA | 550.323771 | 766.713415 | 19 | 83 | 861 | 86 | 10969 | 1659700 | 1316795 | Normal | Normal |
| Abbyville | KS | 255.819516 | 225.96319 | 2 | 27 | 265 | 3 | 466 | 477452 | 55927 | Normal | Normal |
| Abercrombie | MN | 249.071722 | 672.245875 | 16 | 127 | 1307 | 0 | 9 | 1414897 | 1081 | Normal | Normal |
| Abercrombie | ND | 498.143444 | 1344.49175 | 16 | 127 | 1307 | 0 | 9 | 1414897 | 1081 | Normal | Normal |
| Aberdeen | ID | 2879.32878 | 3731.12462 | 47 | 245 | 3164 | 146 | 20693 | 3902778 | 2484084 | Normal | Normal |
| Aberdeen | MS | 21390.3272 | 50693.3465 | 143 | 4049 | 38390 | 119 | 16486 | 64354095 | 1978975 | Extremely High | Normal |
| Aberdeen | NJ | 6221.92 | 10168.81 | 266 | 4223 | 36092 | 47 | 117135 | 69555106 | 14061141 | Extremely High | Extremely High |

## Part 2 – Histograms and Correlations

### Part 2.1 – Histograms

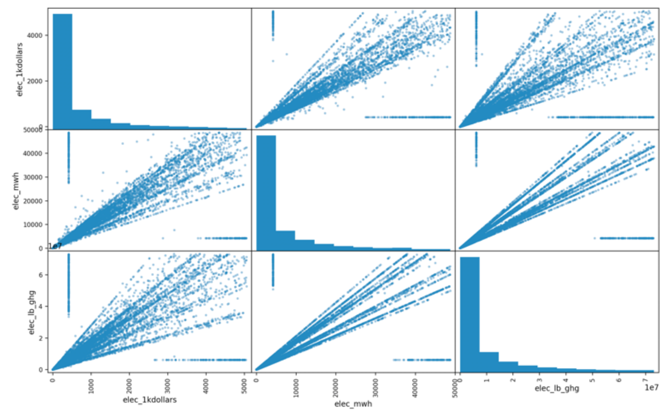
In this part I choose 9 attributes to plot the histograms: 'elec_score', 'gas_score', 'num_establishments', 'elec_1kdollars', 'elec_mwh', 'gas_1kdollars', 'gas_mcf', 'elec_lb_ghg', 'gas_lb_ghg'. The result is shown below:




From this result, I find that all attributes' histograms are left-skewed, which indicates that although I have already handled the maximum value before, there are still have many huge numbers in it. I think it is acceptable because some cities can have high-level energy usage and it is an exact problem this project wants to analyze and predict.

### Part 2.2 – Correlations

In this part, I choose three attributes: elec_1ldollars, elec_mwh, and elec_lb_ghg. And I draw a set of scatter plot to show these attributes relationship.




From the result, we can see that these attributes are related to each other because their scatter plot are not randomly distributed. In the contrary their points are dense into multiple straight lines of emission patterns. It may indicate that the three attributes may have multiple linear correlation.

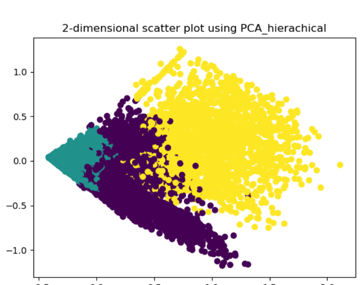
## Part 3 – Cluster Analysis

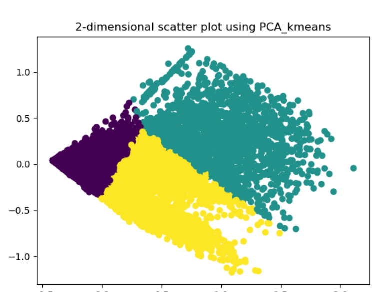
In this part, I use hierarchical clustering, k-means and dbscan clustering to complete cluster analysis. At first, I transform the non-numeric attribute to numeric and then I normalize the dataset. Before I start the cluster analysis, I also drop four attributes: city, state, elec_class, and gas_class, since city and state are keys of the record while elec_class and gas_class are derived from attributes elec_mwh and gas_mcf. These attributes will interfere with the cluster results.

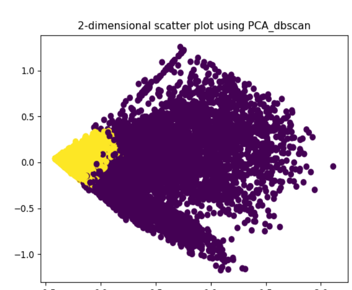
The hierarchical clustering, k-means and dbscan clustering analysis 9 attributes: elec_score, gas_score, num_establishments, elec_1kdollars, elec_mwh, gas_1kdollars, gas_mcf, elec_lb_ghg, and gas_lb_ghg. The results are shown below:

| | silhouette_score | calinski_harabaz_score |
|---|---|---|
| **hierarchical clustering** | 0.5272252264789483 | 10343.484836067191 |
| **k-means** | 0.5812689457864456 | 11283.33168500775 |
| **dbscan clustering** | 0.5509594204732038 | 12151.140160750663 |

For the silhouette_score, k-means performs best since higher silhouette_score is better while hierarchical clustering performs worst. And dbscan performs best for the calinski_harabaz_score. I also use the PCA algorithm to reduce the dimensions and plot the 2D scatter plot. The results are shown below:



From the result, I find that most data are compact and three clustering result are different. K-means and hierarchical clustering have three clusters. K-means has the best clustering result, while hierarchical clustering's one cluster points appear in the other cluster. Dbscan clustering has only 2 clusters and I think these 2 clusters are not balanced.

## Part 4 – Association Rules / Frequent Itemset Mining Analysis

In this part, I select only a few attributes to finish this analysis: state, elec_class, and gas_class. The reason why I delete the city is that there are about 10000 different city name and it will be tough to analysis. The number of states in this commercial dataset is 46, which is acceptable. Since elec_class and gas_class have same label name: Normal, High, and Extremely High, I add elec_class or gas_class before the label to distinguish them. Now the data frame is like:

```
  state_abbr                  elec_class                  gas_class
0         LA  elec_class_Extremely_High  gas_class_Extremely_High
1         ME          elec_class_Normal          gas_class_Normal
2         WI          elec_class_Normal          gas_class_Normal
3         PA          elec_class_Normal          gas_class_Normal
4         TX          elec_class_Normal          gas_class_Normal
5         PA          elec_class_Normal          gas_class_Normal
6         KS          elec_class_Normal          gas_class_Normal
7         MN          elec_class_Normal          gas_class_Normal
8         ND          elec_class_Normal          gas_class_Normal
9         ID          elec_class_Normal          gas_class_Normal
```

After that, I encode this data frame value, train the model, and get the results. I use three different support levels: 0.1, 0.06, 0.04 to get the frequent items. I also set the min_confidence to 0.7 to get the association rules. The sorted results are shown below:

For min_support = 0.1:

```
for min_support = 0.1
       support                             itemsets
3 0.8063681715                   (gas_class_Normal)
1 0.7820410174                  (elec_class_Normal)
4 0.7118310033  (elec_class_Normal, gas_class_Normal)
0 0.1572334791                    (elec_class_High)
2 0.1353082466                     (gas_class_High)
           antecedents          consequents      support   confidence
0  (elec_class_Normal)   (gas_class_Normal) 0.7118310033 0.9102220822
1   (gas_class_Normal)  (elec_class_Normal) 0.7118310033 0.8827617811
```

For min_support = 0.6:

```
for min_support = 0.06
        support                              itemsets
7  0.8063681715                    (gas_class_Normal)
5  0.7820410174                   (elec_class_Normal)
10 0.7118310033  (elec_class_Normal, gas_class_Normal)
4  0.1572334791                     (elec_class_High)
6  0.1353082466                      (gas_class_High)
1  0.0672538030                                  (PA)
8  0.0666995135      (gas_class_High, elec_class_High)
0  0.0654061711                                  (IL)
2  0.0649134692                                  (TX)
9  0.0648518815    (elec_class_High, gas_class_Normal)
3  0.0607255035           (elec_class_Extremely_High)
           antecedents          consequents      support   confidence
0  (elec_class_Normal)   (gas_class_Normal) 0.7118310033 0.9102220822
1   (gas_class_Normal)  (elec_class_Normal) 0.7118310033 0.8827617811
2     (gas_class_High)    (elec_class_High) 0.0666995135 0.4929449249
3    (elec_class_High)     (gas_class_High) 0.0666995135 0.4242068155
4    (elec_class_High)   (gas_class_Normal) 0.0648518815 0.4124559342
5   (gas_class_Normal)    (elec_class_High) 0.0648518815 0.0804246544
```

For min_support = 0.5:

```
for min_support = 0.05
        support                              itemsets
9  0.8063681715                    (gas_class_Normal)
6  0.7820410174                   (elec_class_Normal)
16 0.7118310033  (elec_class_Normal, gas_class_Normal)
5  0.1572334791                     (elec_class_High)
8  0.1353082466                      (gas_class_High)
2  0.0672538030                                  (PA)
13 0.0666995135      (gas_class_High, elec_class_High)
0  0.0654061711                                  (IL)
3  0.0649134692                                  (TX)
14 0.0648518815    (elec_class_High, gas_class_Normal)
4  0.0607255035            (elec_class_Extremely_High)
1  0.0588778715                                  (NY)
7  0.0583235819             (gas_class_Extremely_High)
15 0.0543819671    (gas_class_High, elec_class_Normal)
11 0.0532118002                (IL, gas_class_Normal)
12 0.0524727474                (TX, gas_class_Normal)
10 0.0509946419               (IL, elec_class_Normal)
           antecedents          consequents      support   confidence
0  (elec_class_Normal)   (gas_class_Normal) 0.7118310033 0.9102220822
1   (gas_class_Normal)  (elec_class_Normal) 0.7118310033 0.8827617811
7                 (IL)   (gas_class_Normal) 0.0532118002 0.8135593220
8                 (TX)   (gas_class_Normal) 0.0524727474 0.8083491461
9                 (IL)  (elec_class_Normal) 0.0509946419 0.7796610169
2     (gas_class_High)    (elec_class_High) 0.0666995135 0.4929449249
3    (elec_class_High)     (gas_class_High) 0.0666995135 0.4242068155
4    (elec_class_High)   (gas_class_Normal) 0.0648518815 0.4124559342
6     (gas_class_High)  (elec_class_Normal) 0.0543819671 0.4019116978
5   (gas_class_Normal)    (elec_class_High) 0.0648518815 0.0804246544
```
From the result I can know that:

1. Most states have normal gas usage.
2. Most states have normal electricity usage, but the number of them is less than the number of normal gas usage states.
3. Most states have both normal electricity usage and normal gas usage.
4. Illinois(IL) state has normal electricity usage and normal gas usage.
5. Texas(TX) state has normal gas usage.

## Part 5 – Hypothesis Testing & Classification

### Part 5.1 Hypothesis Testing

In this part, I write three hypotheses:

1. The mean difference between attribute elec_score and attribute gas_score are not significant.
2. Attribute elec_1kdollars has significant influence on attribute elec_class.
3. Attribute elec_score, num_establishment, elec_1kdollars, elec_1b_ghg have significant influence on attribute elec_class.

For hypothesis 1, I use the t-test to verify because t-test generally only compares two sets of data to see if there is any significance of the difference. The result are as followed:

```
LeveneResult(statistic=0.6073946322876103, pvalue=0.43577614667793096)
Ttest_indResult(statistic=-0.7221660570830193, pvalue=0.47019758960573177)
```

The p-value in LeveneResult of these two attributes is larger than 0.05, which means these two attributes have homogeneity of variance. Set equal_val to True and get the p-value in Ttest_indResult is also larger than 0.05, which means hypothesis 1 is true: The mean difference between attribute elec_score and attribute gas_score are not significant.

For hypothesis 2, I use ANOVA to verify because it tests the significance of changes in one variable on changes in another. The result are as followed:

```
                         df          sum_sq        mean_sq                 F       PR(>F)
elec_1kdollars     1.0000000000 812.1396469074 812.1396469074 26537.4383094791 0.0000000000
Residual       16235.0000000000 496.8485282482   0.0306035435              nan          nan
```

df represents freedom; Sum_sq represents sum of squares; Mean_sq means mean the sum of square; F represents the value of F test statistic; PR(>F) represents the test p-value. The p-value of this hypothesis is 0.00000, which means that the result is highly significant. Therefore, hypothesis 2 is true: Attribute elec_1kdollars has a significant influence on attribute elec_class.

For hypothesis 3, I use the logistic regression model to verify because it is suitable for multi-variance and multinomial distribution. The result are as followed:

```
Optimization terminated successfully.
         Current function value: 0.624448
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                12989
Model:                          Logit   Df Residuals:                    12985
Method:                           MLE   Df Model:                            3
Date:                Sun, 03 Nov 2019   Pseudo R-squ.:                 -0.9114
Time:                        15:55:25   Log-Likelihood:                -8111.0
converged:                       True   LL-Null:                       -4243.4
Covariance Type:            nonrobust   LLR p-value:                     1.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1             2.4066      0.142     16.966      0.000       2.129       2.685
x2             4.0309      0.197     20.416      0.000       3.644       4.418
x3            -3.1781      0.205    -15.492      0.000      -3.580      -2.776
x4            -1.6014      0.169     -9.462      0.000      -1.933      -1.270
==============================================================================
```

Look at the column P > |z|, this column represents the p-value. Since all attribute's p-value is smaller than 0.001, I can think that attribute elec_score, num_establishment, elec_1kdollars, elec_1b_ghg have significant influence on attribute elec_class. Therefore, hypothesis 3 is true.

**Part 5.2 Classification**

In this part I use column 'elec_score', 'gas_score', 'num_establishments', 'elec_1kdollars', 'elec_mwh', 'gas_1kdollars', 'gas_mcf', 'elec_lb_ghg', and 'gas_lb_ghg' as training attribute and use column 'elec_class' as electricity prediction class and use column 'gas_class' as gas prediction class.

I use five kinds of classifiers to do the classification work: decision tree, kNN, Naïve Bayes, SVM, Random Forest. Here are the results:

For electricity classification:

```
## CART ##
The accuracy of train set for  CART: 0.956963 (0.006557)
The accuracy of validate set is  0.958435960591133
[[ 156   19   31]
 [  17 2520   20]
 [  25   23  437]]
              precision    recall  f1-score   support

         0.0       0.79      0.76      0.77       206
         1.0       0.98      0.99      0.98      2557
         2.0       0.90      0.90      0.90       485

    accuracy                           0.96      3248
   macro avg       0.89      0.88      0.89      3248
weighted avg       0.96      0.96      0.96      3248
```

```
## KNN ##
The accuracy of train set for  KNN: 0.947648 (0.004744)
The accuracy of validate set is  0.9507389162561576
[[ 164   12   30]
 [  23 2499   35]
 [  23   37  425]]
              precision    recall  f1-score   support

         0.0       0.78      0.80      0.79       206
         1.0       0.98      0.98      0.98      2557
         2.0       0.87      0.88      0.87       485

    accuracy                           0.95      3248
   macro avg       0.88      0.88      0.88      3248
weighted avg       0.95      0.95      0.95      3248
```

```
## BAYE ##
The accuracy of train set for  BAYE: 0.878821 (0.007587)
The accuracy of validate set is  0.8796182266009852
[[ 174    2   30]
 [  58 2346  153]
 [  64   84  337]]
              precision    recall  f1-score   support

         0.0       0.59      0.84      0.69       206
         1.0       0.96      0.92      0.94      2557
         2.0       0.65      0.69      0.67       485

    accuracy                           0.88      3248
   macro avg       0.73      0.82      0.77      3248
weighted avg       0.89      0.88      0.88      3248
```

```
## SVM ##
The accuracy of train set for  SVM: 0.930865 (0.004690)
The accuracy of validate set is  0.9322660098522167
[[ 125   32   49]
 [  19 2500   38]
 [  26   56  403]]
              precision    recall  f1-score   support

         0.0       0.74      0.61      0.66       206
         1.0       0.97      0.98      0.97      2557
         2.0       0.82      0.83      0.83       485

    accuracy                           0.93      3248
   macro avg       0.84      0.81      0.82      3248
weighted avg       0.93      0.93      0.93      3248
```

```
## RandomForest ##
The accuracy of train set for  RandomForest: 0.964201 (0.003645)
The accuracy of validate set is  0.9612068965517241
[[ 169   15   22]
 [  22 2517   18]
 [  22   27  436]]
              precision    recall  f1-score   support

         0.0       0.79      0.82      0.81       206
         1.0       0.98      0.98      0.98      2557
         2.0       0.92      0.90      0.91       485

    accuracy                           0.96      3248
   macro avg       0.90      0.90      0.90      3248
weighted avg       0.96      0.96      0.96      3248
```

The random forest classifier performs best because its accuracy for training set and validation set is highest(0.96). The Bayes classifier performs worst and its accuracy is 0.87.

For gas classification:

```
## CART ##
The accuracy of train set for  CART: 1.000000 (0.000000)
The accuracy of validate set is  1.0
[[ 192    0    0]
 [   0 2588    0]
 [   0    0  468]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       192
         1.0       1.00      1.00      1.00      2588
         2.0       1.00      1.00      1.00       468

    accuracy                           1.00      3248
   macro avg       1.00      1.00      1.00      3248
weighted avg       1.00      1.00      1.00      3248
```

```
## KNN ##
The accuracy of train set for  KNN: 0.980676 (0.002786)
The accuracy of validate set is  0.9802955665024631
[[ 176    0   16]
 [   0 2571   17]
 [   6   25  437]]
              precision    recall  f1-score   support

         0.0       0.97      0.92      0.94       192
         1.0       0.99      0.99      0.99      2588
         2.0       0.93      0.93      0.93       468

    accuracy                           0.98      3248
   macro avg       0.96      0.95      0.95      3248
weighted avg       0.98      0.98      0.98      3248
```

```
## BAYE ##
The accuracy of train set for  BAYE: 0.927632 (0.007965)
The accuracy of validate set is  0.9356527093596059
[[ 185    0    7]
 [   0 2421  167]
 [  26    9  433]]
              precision    recall  f1-score   support

         0.0       0.88      0.96      0.92       192
         1.0       1.00      0.94      0.96      2588
         2.0       0.71      0.93      0.81       468

    accuracy                           0.94      3248
   macro avg       0.86      0.94      0.90      3248
weighted avg       0.95      0.94      0.94      3248
```

```
## SVM ##
The accuracy of train set for  SVM: 0.983139 (0.003811)
The accuracy of validate set is  0.9833743842364532
[[ 184    0    8]
 [   0 2585    3]
 [   7   36  425]]
              precision    recall  f1-score   support

         0.0       0.96      0.96      0.96       192
         1.0       0.99      1.00      0.99      2588
         2.0       0.97      0.91      0.94       468

    accuracy                           0.98      3248
   macro avg       0.97      0.96      0.96      3248
weighted avg       0.98      0.98      0.98      3248
```

```
## RandomForest ##
The accuracy of train set for  RandomForest: 0.999692 (0.000511)
The accuracy of validate set is  1.0
[[ 192    0    0]
 [   0 2588    0]
 [   0    0  468]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       192
         1.0       1.00      1.00      1.00      2588
         2.0       1.00      1.00      1.00       468

    accuracy                           1.00      3248
   macro avg       1.00      1.00      1.00      3248
weighted avg       1.00      1.00      1.00      3248
```

The decision tree classifier performs best because its accuracy for training set and validation set is highest(1.0). The Bayes classifier performs worst and its accuracy is 0.93. The whole accuracy of gas classification is higher than electricity classification.