Healthcare Fraud Detection Project

Yiheng Shen

Table of Contents

Introduction *2*
Data Cleaning *2*
Feature Engineering & Exploratory Data Analysis *3*
K-means Clustering + PCA *7*
encoding and standardizing 7
k-means clustering 7
PCA 8
Anomaly (Outliers) Detection 8
Unsupervised KNN *9*
Hyper-Parameters 9
Anomaly (Outliers) Detection 11
Unsupervised Autoencoder *12*
Combinations of hidden neurons 12
Anomaly (Outliers) Detection 13
Isolation Forest *14*
Anomaly (Outliers) Detection 14
Conclusion *15*

Introduction

Medical fraud does exist in United States. For example, a fraudster physician may collude with a pharmacist to add more expensive medicines to the prescription claim without the awareness of the patient. This report tried to detect the hospitals which overcharge patients in US using these two datasets: National Summary of Inpatient Charge Data by Medicare Severity Diagnosis Related Group (MS-DRG), FY2015 and Median Household Income by State: 2015 derived from Bureau of Economic Analysis. Detailed description of these two datasets could be found via the hyper-link.

Data Cleaning

The first dataset contained 12 columns as below, where DRG represents diagnosis-related group, Total Discharges meant the number of discharges billed by the provider for inpatient hospital services, and Covered Charges meant charges for covered services that your health plan paid for.

#	Column	Non-Null Count	Dtype
0	DRG Definition	163065 non-null	object
1	Provider Id	163065 non-null	int64
2	Provider Name	163065 non-null	object
3	Provider Street Address	163065 non-null	object
4	Provider City	163065 non-null	object
5	Provider State	163065 non-null	object
6	Provider Zip Code	163065 non-null	int64
7	Hospital Referral Region Description	163065 non-null	object
8	Total Discharges	163065 non-null	int64
9	Average Covered Charges	163065 non-null	object
10	Average Total Payments	163065 non-null	object
11	Average Medicare Payments	163065 non-null	object

I first revised the column names of the columns in the dataset and removed the "$" before Average_Covered_Charges, Average_Total_Payments, and Average_Medicare_Payments, and then checked that there were no missing values in this dataset.

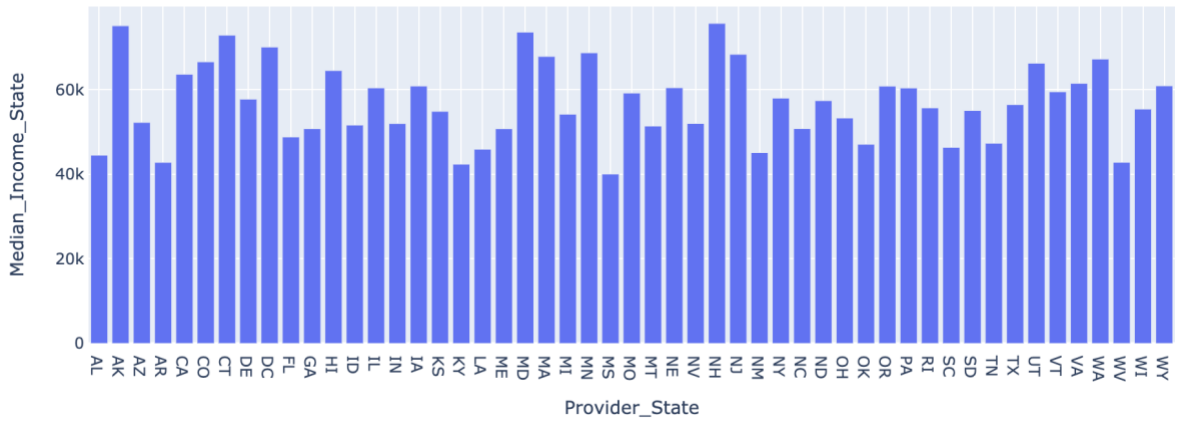
```
# change column names
df.columns = ['DRG_Definition', 'Provider_Id', 'Provider_Name', 'Provider_Street_Address',
      'Provider_City', 'Provider_State', 'Provider_Zip_Code', 'Hospital_Referral_Region_Description',
            'Total_Discharges', 'Average_Covered_Charges', 'Average_Total_Payments', 'Average_Medicare_Payments']
df.head(2)
```

	DRG_Definition	Provider_Id	Provider_Name	Provider_Street_Address	Provider_City	Provider_State	Provider_Zip_Code	Hospital_Referral_Region_Description
0	039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	10001	SOUTHEAST ALABAMA MEDICAL CENTER	1108 ROSS CLARK CIRCLE	DOTHAN	AL	36301	AL - Dothan
1	039 - EXTRACRANIAL PROCEDURES W/O CC/MCC	10005	MARSHALL MEDICAL CENTER SOUTH	2505 U S HIGHWAY 431 NORTH	BOAZ	AL	35957	AL - Birmingham

```
#reomve $
df['Average_Covered_Charges'] = df['Average_Covered_Charges'].str.replace('$', '').astype(float)
df['Average_Total_Payments'] = df['Average_Total_Payments'].str.replace('$', '').astype(float)
df['Average_Medicare_Payments'] = df['Average_Medicare_Payments'].str.replace('$', '').astype(float)
```

Then I merged the second dataset, the data of Median Household Income by State to the first dataset using left join on Provider_State so as to create features for modelling. The bar plot below showed the distribution of median income across all states in US.

```
fig = px.bar(income, x="Provider_State", y="Median_Income_State",width=1000, height=400)
fig.show()
```

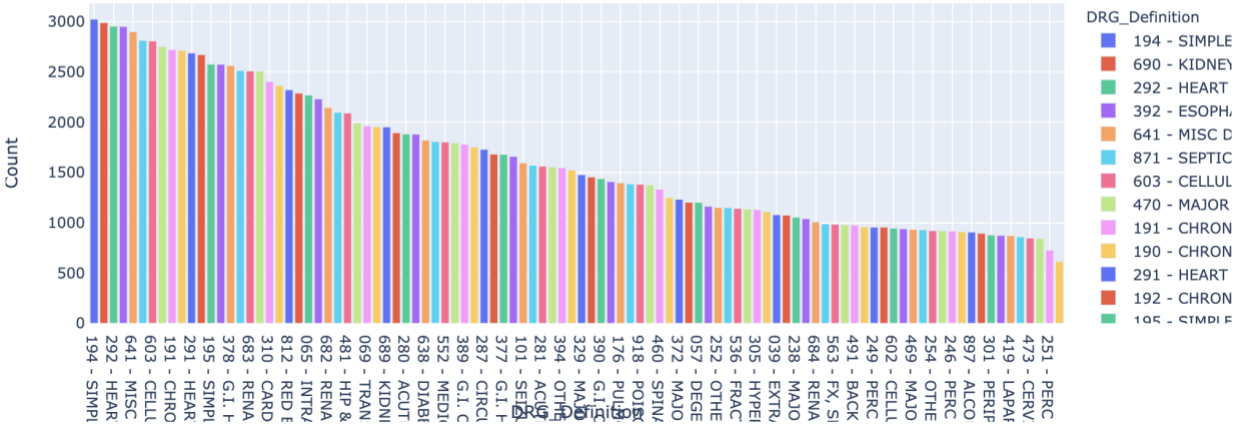


Feature Engineering & Exploratory Data Analysis

EDA was conducted next to help find the features the anomaly detection models would need. I first counted the number of DRGs and visualized its distribution. Some DRGs had much

more recordings than other DRGs. Hence it is necessary to create two groups of features: one group with specific DRGs and the other without specific DRGs.

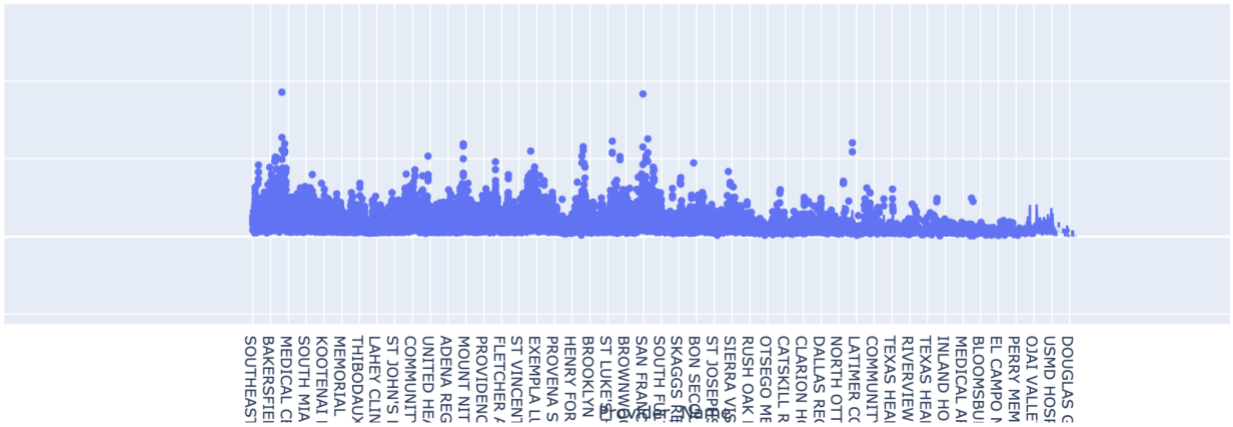
```
# count DRG_Definition
df_count = df['DRG_Definition'].value_counts()
df_count = pd.DataFrame(df_count).reset_index()
df_count.columns = ['DRG_Definition','Count']

fig = px.bar(df_count, x='DRG_Definition', y='Count', color = 'DRG_Definition', width=1500, height=400)
fig.show()
```



Codes to create the features were too much to be attached in this report. The features created for anomaly detection include:

1. **average covered charge of a hospital on all DRG**

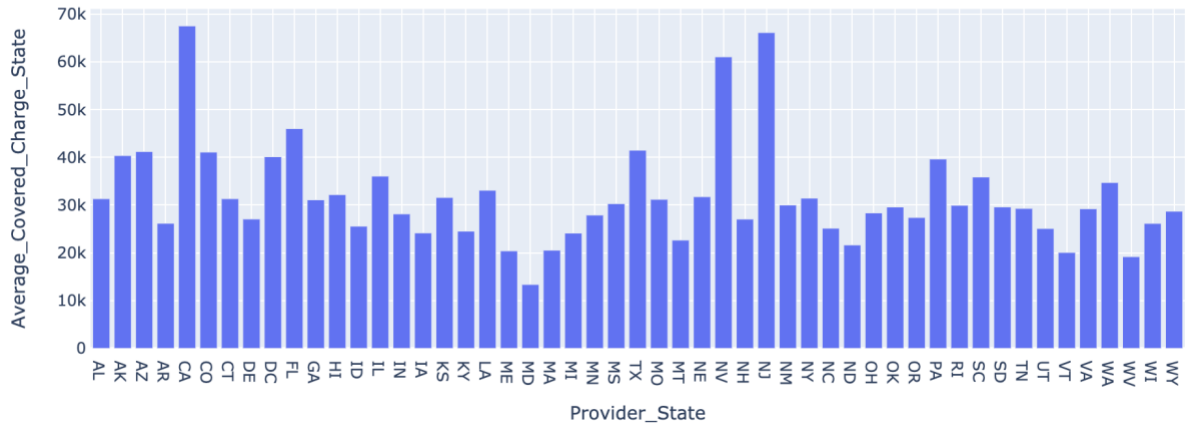
```
# visualization
fig = px.box(df, x="Provider_Name", y="Average_Covered_Charges",width=1500, height=400)
fig.show()
```



This boxplot shows certain hospitals charge very high on certain DRG, but which specific DRG they charge higher on were not known yet.

2. **average covered charge on a DRG**
3. **average covered charges of a city on a DRG**
4. **average covered charges of a state on a DRG**

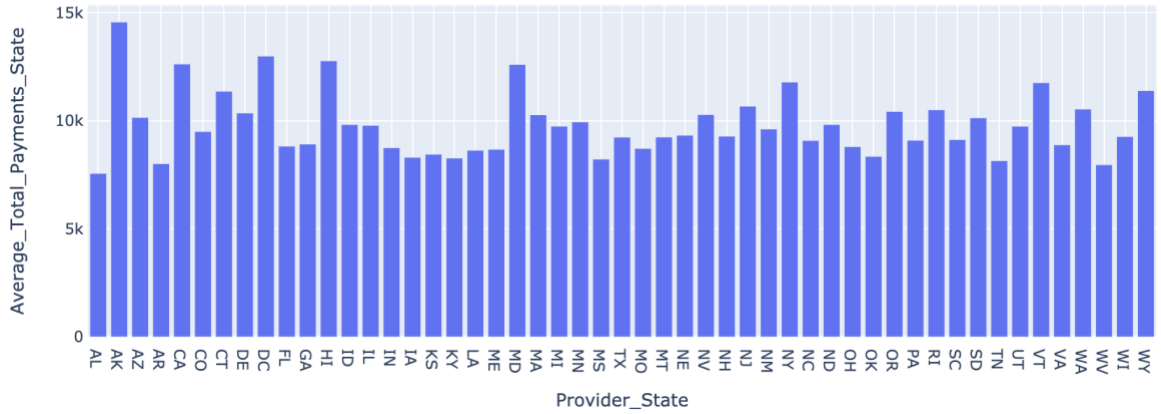
```
# visualization
df_selected = df4[['Provider_State', 'Average_Covered_Charge_State']]
df_selected = df_selected.drop_duplicates()
fig = px.bar(df_selected, x="Provider_State", y="Average_Covered_Charge_State",width=1000, height=400)
fig.show()
```



This histogram showed CA, NV and NJ overall charges higher.

5. **5: median income of a state**
6. **average covered charges versus the median income of a state**
7. **average covered charges on a DRG versus the median income of a state**
8. **average covered charges of a hospital on all DRG versus the median income of a state**
9. **average total payments of a hospital on all DRG**
10. **average total payments of a DRG**
11. **average total payments of a city on a DRG**
12. **average total payments of a state on a DRG**

```
# visualization
df_selected = df12[['Provider_State', 'Average_Total_Payments_State']]
df_selected = df_selected.drop_duplicates()
fig = px.bar(df_selected, x="Provider_State", y="Average_Total_Payments_State",width=1000, height=400)
fig.show()
```



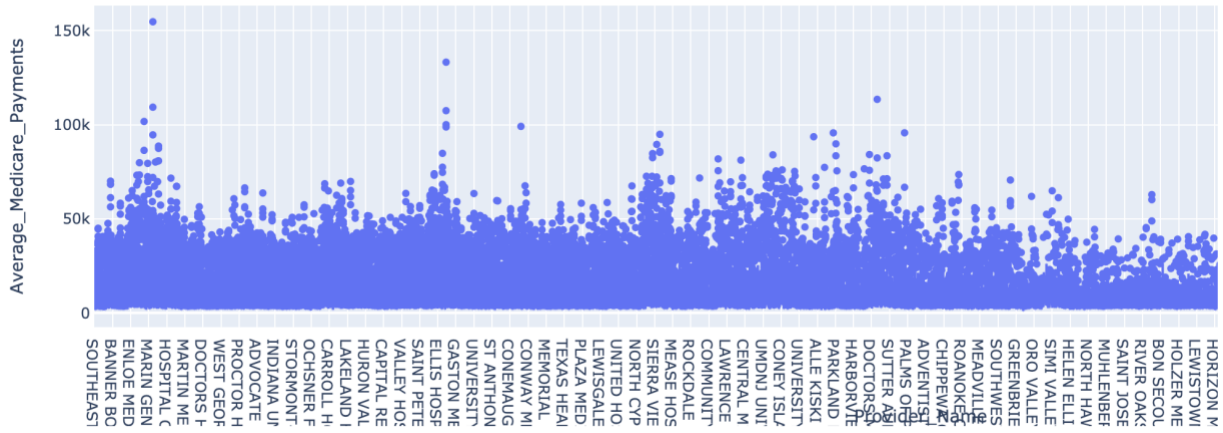
13. average total payments versus the median income of a state

14. average total payments on a DRG versus the median income of a state¶

15. average total payments of a hospital on all DRG versus the median income of a state

16. average medicare payments of a hospital on all DRG¶

```
# visualization
fig = px.box(df16, x="Provider_Name", y="Average_Medicare_Payments",width=1500, height=400)
fig.show()
```



17. average medicare payments on a DRG

18. average medicare payments of a city on a DRG

19. **average medicare payments of a state on a DRG**
20. **average medicare payments versus the median income of a state**
21. **average medicare payments on a DRG versus the median income of a state**
22. **average medicare payments of a hospital on all DRG versus the median income of a state**

K-means Clustering + PCA

encoding and standardizing

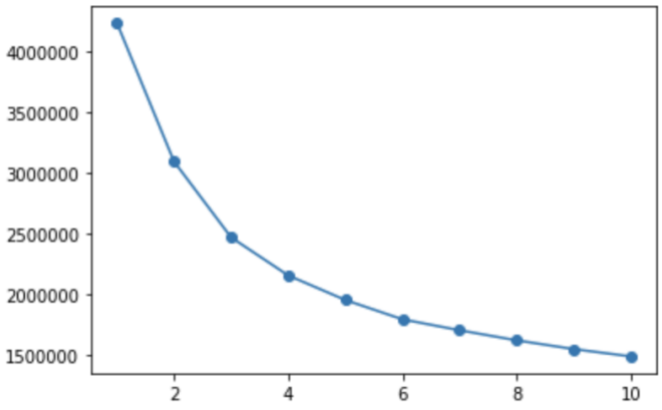
OneHotEncoding was conducted on Provider_State as it was a categorical variable and standard scaling was used on numeric variables.

```
#one hot encoder on df21
ohe = OneHotEncoder(sparse = False)
category = ohe.fit_transform(df21[['Provider_State']])
category = pd.DataFrame(category,index = df.index, columns = ohe.get_feature_names(),dtype = np.int8)
```

```
#standardize
numeric = df21[df21.columns[8:33]]
std = StandardScaler().fit_transform(numeric)
numeric = pd.DataFrame(std)
numeric.columns = df21.columns[8:33].values
data = pd.concat([category,numeric],axis=1)
```

k-means clustering

The elbow plot was used to find the best number of clusters, which was 6.



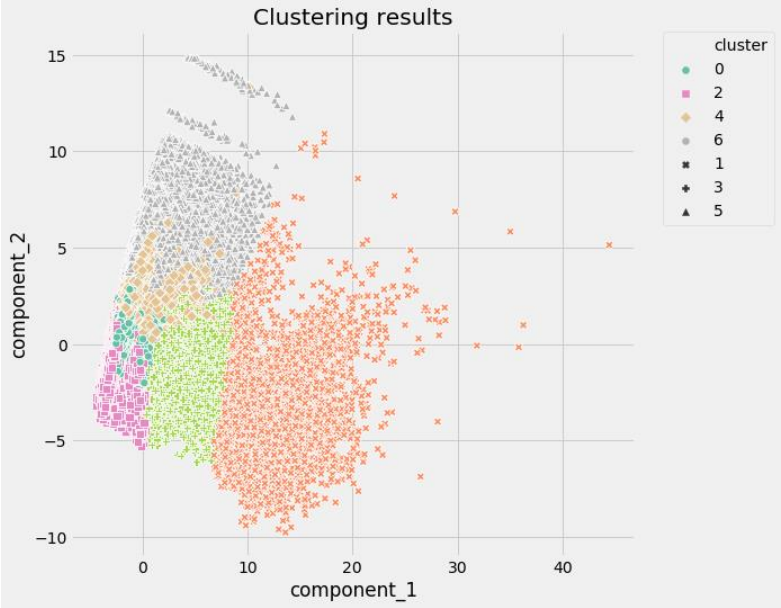
The number of samples in each cluster was as below:

```
num = pd.Series(kmeans.labels_).value_counts()
num
```

```
2    55219
0    43772
3    26142
5    17466
4    14640
1     5826
dtype: int64
```

PCA

Principal component analysis was conducted to reduce the number of features to 2 for visualization.



The result of PCA showed the clusters of hospitals. This plot showed that cluster 1 had many outliers.

Anomaly (Outliers) Detection

The 4 feaures: Total_Discharges, Average_Covered_Charges, Average_Total_Payments, and Average_Medicare_PaymentsSecondly were selected to be the metrics of charging. Table below showed that cluster 1 on average charged higher than other clusters did.

```
center = pd.DataFrame(kmeans.cluster_centers_)
center.columns = data.columns.values
r = pd.concat([center, num], axis = 1)
```

```
r.iloc[:, 51:55]
```

	Total_Discharges	Average_Covered_Charges	Average_Total_Payments	Average_Medicare_Payments
0	0.104947	-0.235915	-0.447458	-0.453584
1	-0.298755	3.022823	3.799586	3.789235
2	-0.177282	-0.546855	-0.461697	-0.453833
3	0.135030	0.606845	0.715221	0.688803
4	0.265817	-0.342311	0.186626	0.201740
5	-0.028314	0.688903	0.085410	0.106265

Therefore, hospitals in cluster 1 were the hospitals that overcharged. There were 2419 hospitals in cluster 1, which was a too large number, so I only choose hospitals with more than 30 anomalous recordings as overcharging hospitals and the number of hospitals reduced to 40.

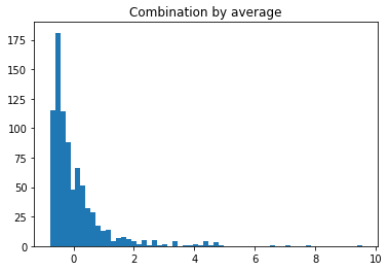
Unsupervised KNN

Hyper-Parameters

Unsupervised KNN was used for anomaly detection as well. Hyper-parameter was first used to find the best parameters for this dataset. 4 methods of boundary determination were then used to determine the boundary in KNN.

```
#train kNN detector
from pyod.models.knn import KNN
from pyod.models.combination import aom, moa, average, maximization
from pyod.utils.utility import standardizer

n_clf = 20
k_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110,
 120, 130, 140, 150, 160, 170, 180, 190, 200]

train_scores = np.zeros([X_train.shape[0], n_clf])
test_scores = np.zeros([X_test.shape[0], n_clf])
train_scores.shape

# Modeling
for i in range(n_clf):
    k = k_list[i]
    clf = KNN(n_neighbors=k, method='largest')
    clf.fit(X_train)

# Store the results in each column:
    train_scores[:, i] = clf.decision_scores_
    test_scores[:, i] = clf.decision_function(X_test)

# Decision scores have to be normalized before combination
train_scores, test_scores = standardizer(train_scores, test_scores)
```

1. **Average Method Determination of Boundary**

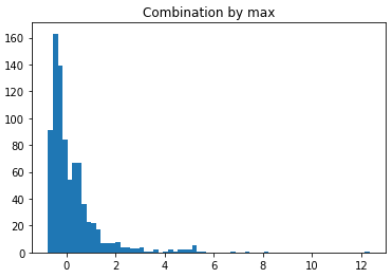


2 was used as the threshold. The number of "1" under this method was 41, accounting for 4.91% of total samples.

```
#from the plot I think 2 will be a good threshold
df_test = pd.DataFrame(X_test)
df_test['y_by_average_score'] = y_by_average
df_test['y_by_average_cluster'] = np.where(df_test['y_by_average_score'] < 2, 0, 1)
df_test['y_by_average_cluster'].value_counts()
```

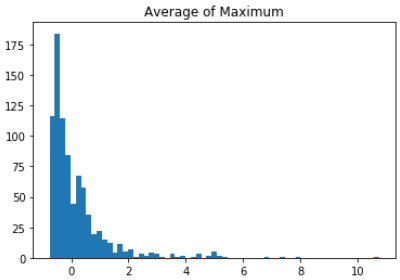
```
0    794
1     41
Name: y_by_average_cluster, dtype: int64
```

2. **Maximum of Maximum Method Determination of Boundary**



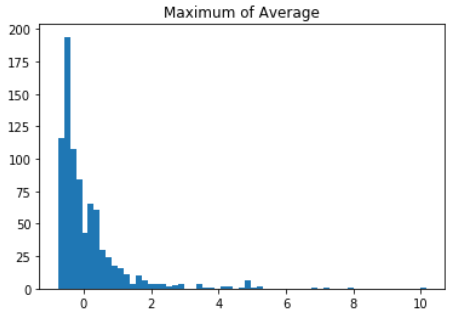
The number of "1" under this method was 50, 6.02% of total.

3. **The Average of Maximum Method Determination of Boundary**



The number of "1" under this method was 44, 5.29% of total.

4. **The Maximum of Average Method Determination of Boundary**



The number of "1" under this method was 42, 5.05% of total.

Anomaly (Outliers) Detection

Four tables below show that Average Method is the best method for anomaly detection of this dataset, as the difference between the scores of cluster 0 and cluster 1 is the largest. So I attached the labels generated from Average Method back to the dataset.

```
df_test.groupby('y_by_average_cluster').mean().iloc[:,76:77]
```

y_by_average_cluster	y_by_average_score
0	-0.111068
1	3.734680

```
df_test.groupby('y_by_average_cluster').mean().iloc[:,78:79]
```

y_by_average_cluster	y_by_maximization_cluster
0	0.011335
1	1.000000

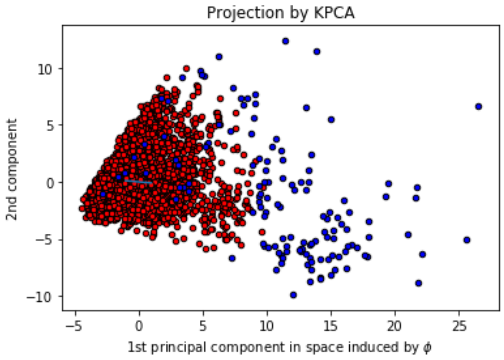
```
df_test.groupby('y_by_average_cluster').mean().iloc[:,80:81]
```

y_by_average_cluster	y_by_aom_cluster
0	0.003778
1	1.000000

```
df_test.groupby('y_by_average_cluster').mean().iloc[:,82:83]
```

y_by_average_cluster	y_by_moa_cluster
0	0.001259
1	1.000000

Result of linear kernel PCA also indicated that Average Method well captured the outliers.



Average Method gave 133 hospitals who overcharged and this list of hospitals overlapped with the list given by K-means Clustering.

Unsupervised Autoencoder

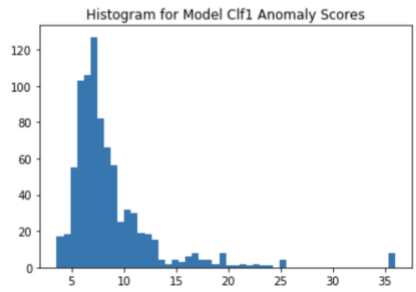
These features was used as the benchmark for autoencoder: Average_Covered_Charges, Average_Covered_Charge_vs_Income_Ratio, Average_Total_Payments, Average_Total_Payments_by_Income_Ratio, Average_Medicare_Payments, and Average_Medicare_Payments_by_Income_Ratio.

Combinations of hidden neurons

1. **clf1：hidden_neurons = [25, 2, 2, 25]**

The input layer and the output layer had 25 neurons each. There were two hidden layers, each has two neurons. 15 was chosen as the threshold.

```
import matplotlib.pyplot as plt
plt.hist(y_test_scores, bins='auto')  # arguments are passed to np.histogram
plt.title("Histogram for Model Clf1 Anomaly Scores")
plt.show()
```



2. **clf2：hidden_neurons = [25, 10, 2, 10, 25]**

The input layer and the output layer had 25 neurons each. There were 3 hidden layers with 10, 2, and 10 neurons respectively.

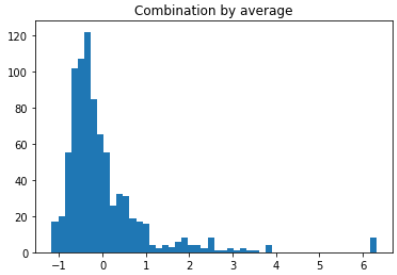
3. **clf3：hidden_neurons = [25, 15, 10, 2, 10,15, 25]**

The input layer and the output layer had 25 neurons each. There were five hidden layers with 15, 10, 2, 10, 15 neurons respectively.

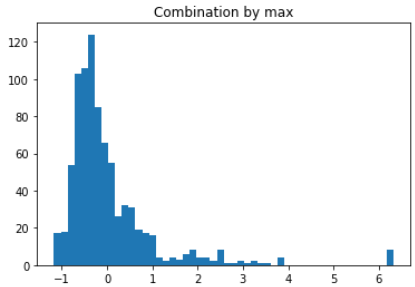
Anomaly (Outliers) Detection

The result of 3 combinations were aggregated. Average method and maximum of maximum method were used to identify the outliers.

1. **Average Method**



2. **Maximum of Maximum Method**



The result of Average Method is slightly better than that of Maximization Method. Hence the list of overcharging hospitals was generated by the Average Method. This method gave 180 hospitals that overcharged.

```
df_test.groupby('y_by_average_cluster').mean()
```

	Average_Total_Payments_by_Income_Ratio	Average_Medicare_Payments	Average_Medicare_Payments_by_Income_Ratioe	score	cluster	y_by_average_score
	-0.066148	-0.060872	-0.066320	8.587124	0.744040	-0.166138
	2.023413	2.065939	2.021344	8.760244	0.578947	3.484525

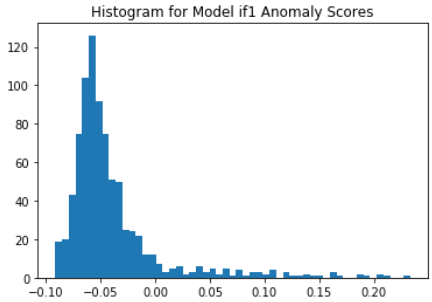
```
df_test.groupby('y_by_maximization_cluster').mean()
```

erage_Total_Payments_by_Income_Ratio	Average_Medicare_Payments	Average_Medicare_Payments_by_Income_Ratioe	score	cluster	y_by_maximization_score
-0.066148	-0.060872	-0.066320	8.587124	0.744040	-0.163056
2.023413	2.065939	2.021344	8.760244	0.578947	3.489383

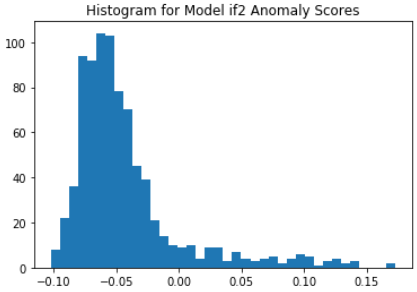
Isolation Forest

Isolation Forest is an anomaly detection algorithm that identifies observations that do not conform to the norm.

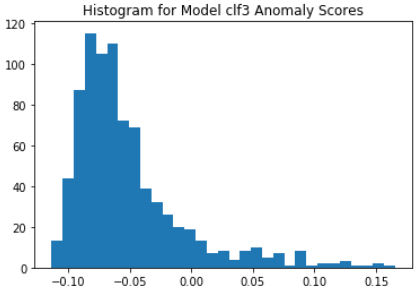
1. **max_samples=100**



2. **max_samples=80**



3. **max_samples=60**



Anomaly (Outliers) Detection

The result of 3 combinations were aggregated. Average method and maximum of maximum method were used to identify the outliers. Average Method was better than Maximum of Maximum Method indicated by all benchmarks. iForest gave 69 overcharging hospitals.

Conclusion

With insights drawn from 4 unsupervised methods: K-means, unsupervised KNN, unsupervised Autoencoder, and iForest, hospitals that overcharged their patients could be captured. Though the number of overcharging hospitals given by the 4 methods were different, the list of hospital names overlapped, which means the combination of the 4 methods might be helpful to fraud detection methods in the healthcare system.