# INFOSYS SPRINGBOARD

**Tushar Bhagat**

# Building Anomaly Detection System using Python

### (Autoencoder approach)

# Problem Statement:

In this project, we delve deep into the thriving sector of **Security** by analyzing a **Anomaly detection on Healthcare Dataset** from a USA-based Health Service Providers, available at the kaggle. This dataset documents all transactions between patients and service providers. Our primary objective is to amplify the efficiency of Healthcare System and avoid fraudulent transactions in **Healthcare system**. We aim to transform the data into a -centric dataset that will facilitate the Base for Anomaly Detection system of patient providing better service , ultimately enhancing security ,efficiency and patient service.

In [1]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from sklearn.preprocessing import StandardScaler
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from tensorflow.keras.utils import plot_model
```

In [2]:
```
X_data_scaled = pd.read_csv(r'C:\Users\tmbha\Downloads\ifosys_springboard\df.csv')
```

In [3]:
```
# Initialize the StandardScaler
scaler = StandardScaler()

# List of columns that don't need to be scaled
columns_to_exclude = ['Name','Full Address']

# List of columns that need to be scaled
columns_to_scale = X_data_scaled.columns.difference(columns_to_exclude)

# Applying the scaler to the necessary columns in the dataset
X_data_scaled[columns_to_scale] = scaler.fit_transform(X_data_scaled[columns_to_scale])

# Dropping identifiers
DropCols = ['Name','Full Address']
X_data_scaled = X_data_scaled.drop(DropCols, axis = 1)

# Display the first few rows of the scaled data
X_data_scaled.head().T
```

Out[3]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **Number of Services** | -0.085301 | -0.025939 | -0.083296 | -0.088109 | -0.082895 |
| **Number of Medicare Beneficiaries** | -0.059308 | 0.076775 | -0.069222 | -0.064716 | -0.059308 |
| **Number of Distinct Medicare Beneficiary/Per Day Services** | -0.070183 | 0.020049 | -0.067135 | -0.074451 | -0.067744 |
| **Average Medicare Allowed Amount** | 0.385450 | 0.086673 | -0.041922 | -0.380709 | -0.291221 |
| **Average Submitted Charge Amount** | -0.046433 | 0.182805 | -0.187794 | -0.328957 | -0.296019 |
| **Average Medicare Payment Amount** | 0.400082 | 0.207649 | -0.064687 | -0.370166 | -0.289505 |
| **Average Medicare Standardized Amount** | 0.414299 | 0.286359 | -0.087154 | -0.372921 | -0.294800 |
| **Diff_submitted_allowed** | -0.166975 | 0.193356 | -0.212261 | -0.282934 | -0.269463 |
| **Is_US** | 0.007746 | 0.007746 | 0.007746 | 0.007746 | 0.007746 |
| **Gender** | -1.560716 | -1.560716 | 0.640731 | 0.640731 | 0.640731 |
| **Entity** | 0.210784 | 0.210784 | 0.210784 | 0.210784 | 0.210784 |
| **Medicare_Participation_Indicator** | 0.017610 | 0.017610 | 0.017610 | 0.017610 | 0.017610 |
| **Place_of_Service** | -1.266985 | 0.789275 | 0.789275 | 0.789275 | 0.789275 |
| **HCPCS_Drug_Indicator** | 0.257051 | 0.257051 | 0.257051 | 0.257051 | 0.257051 |
| **Credentials of the Provider_FrequencyEncoded** | 0.594983 | 0.594983 | -1.684316 | 0.594983 | -1.549260 |
| **City of the Provider_FrequencyEncoded** | 1.571686 | 0.189180 | -0.756245 | 0.702275 | -0.561459 |
| **State Code of the Provider_FrequencyEncoded** | -0.737342 | -0.004973 | -0.989093 | -0.737342 | 1.494517 |
| **ProviderType_FrequencyEncoded** | 1.336743 | -0.940500 | -0.720441 | 1.336743 | 1.336743 |
| **HCPCS Description_FrequencyEncoded** | 0.389268 | -0.450300 | -0.608815 | -0.277448 | -0.060785 |

In [4]:
```
# Define the autoencoder model
input_dim = X_data_scaled.shape[1]

input_layer = Input(shape=(input_dim,))
encoding_layer1 = Dense(32, activation='relu')(input_layer)
encoding_layer2 = Dense(16, activation='relu')(encoding_layer1)
encoded = Dense(input_dim, activation='relu')(encoding_layer1)

decoding_layer1 = Dense(16, activation='relu')(encoded)
decoding_layer2 = Dense(32, activation='relu')(decoding_layer1)
decoder = Dense(input_dim, activation='sigmoid')(decoding_layer2)

autoencoder = Model(inputs=input_layer, outputs=decoder)
autoencoder.compile(optimizer='adam', loss='mse')

# Train the autoencoder on the entire dataset
autoencoder.fit(X_data_scaled, X_data_scaled, epochs=50, batch_size=32, shuffle=True)
```

```
Epoch 1/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 14s 3ms/step - loss: 0.8599
Epoch 2/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 18s 3ms/step - loss: 0.7588
Epoch 3/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - loss: 0.7640
Epoch 4/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - loss: 0.7728
Epoch 5/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - loss: 0.8733
Epoch 6/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - loss: 0.7954
Epoch 7/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 1.0640
Epoch 8/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - loss: 0.7331
Epoch 9/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7776
Epoch 10/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.9520
Epoch 11/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7702
Epoch 12/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7448
Epoch 13/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.8789
Epoch 14/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.6803
Epoch 15/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7201
Epoch 16/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7304
Epoch 17/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.8579
Epoch 18/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 0.8071
Epoch 19/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7380
Epoch 20/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - loss: 0.7224
Epoch 21/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 0.7652
Epoch 22/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 0.7645
Epoch 23/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 0.7692
Epoch 24/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step - loss: 0.6676
Epoch 25/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step - loss: 0.7269
Epoch 26/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - loss: 0.8708
Epoch 27/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 20s 3ms/step - loss: 0.7718
Epoch 28/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 20s 3ms/step - loss: 0.7662
Epoch 29/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 21s 3ms/step - loss: 0.6771
Epoch 30/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7661
Epoch 31/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 13s 4ms/step - loss: 0.7534
Epoch 32/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 19s 3ms/step - loss: 0.7915
Epoch 33/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.8125
Epoch 34/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step - loss: 0.8654
Epoch 35/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.8721
Epoch 36/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7777
Epoch 37/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7483
Epoch 38/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.9695
Epoch 39/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.8095
Epoch 40/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.8479
Epoch 41/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7783
Epoch 42/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 0.7456
Epoch 43/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step - loss: 0.7866
Epoch 44/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7560
Epoch 45/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7175
```

```
Epoch 46/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.8025
Epoch 47/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - loss: 0.8158
Epoch 48/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - loss: 0.8022
Epoch 49/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - loss: 0.7763
Epoch 50/50
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - loss: 0.7161
```

Out[4]: <keras.src.callbacks.history.History at 0x256a94612d0>

In [5]:
```
#displaying summary of model architecture
autoencoder.summary()
```

**Model: "functional_1"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | (None, 19) | 0 |
| dense (Dense) | (None, 32) | 640 |
| dense_2 (Dense) | (None, 19) | 627 |
| dense_3 (Dense) | (None, 16) | 320 |
| dense_4 (Dense) | (None, 32) | 544 |
| dense_5 (Dense) | (None, 19) | 627 |

**Total params:** 8,276 (32.33 KB)
**Trainable params:** 2,758 (10.77 KB)
**Non-trainable params:** 0 (0.00 B)
**Optimizer params:** 5,518 (21.56 KB)

# Inferences from the Summary :

## The autoencoder model consists of the following layers:

- There are total 1000 anomalies are present in data based on the reconstruction error which cross the threshold .
- **Input Layer:** The input layer takes in data with 19 features.
- **Dense Layers:**
  - The first dense layer has 32 units and 640 parameters.
  - The second dense layer has 19 units and 627 parameters.
  - The third dense layer has 16 units and 320 parameters.
  - The fourth dense layer has 32 units and 542 parameters.
  - The final dense layer has 19 units and 627 parameters.
- **Total Parameters:** The model has a total of 8,276 parameters, 2,758 of which are trainable.

In [6]:
```
# Reconstruct the Data and Calculate Reconstruction Error
X_data_reconstructed = autoencoder.predict(X_data_scaled)
reconstruction_errors = np.mean(np.square(X_data_scaled - X_data_reconstructed), axis=1)

#setting threshold
threshold = np.percentile(reconstruction_errors, 99)

# Detect anomalies
anomalies = reconstruction_errors > threshold

print("Anomalies detected:", anomalies)
```

```
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step
Anomalies detected: 0        False
1        False
2        False
3        False
4        False
         ...
99995    False
99996    False
99997    False
99998    False
99999    False
Length: 100000, dtype: bool
```

In [7]:

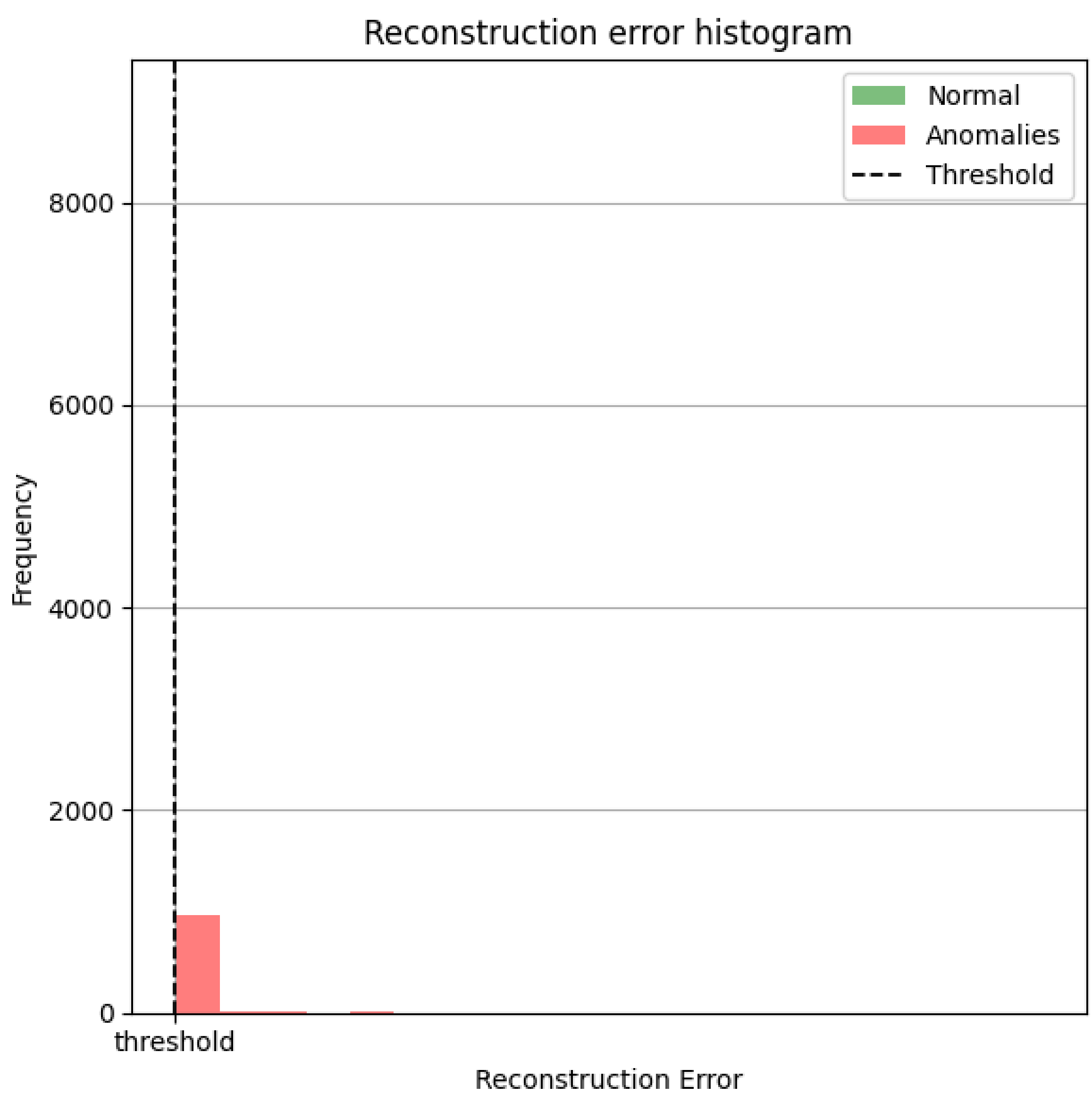
```
reconstruction_errors
```

Out[7]:

```
0        0.266990
1        0.187374
2        0.283561
3        0.071734
4        0.184675
           ...
99995    0.310591
99996    0.604383
99997    0.171216
99998    0.185248
99999    0.188035
Length: 100000, dtype: float64
```

In [8]:

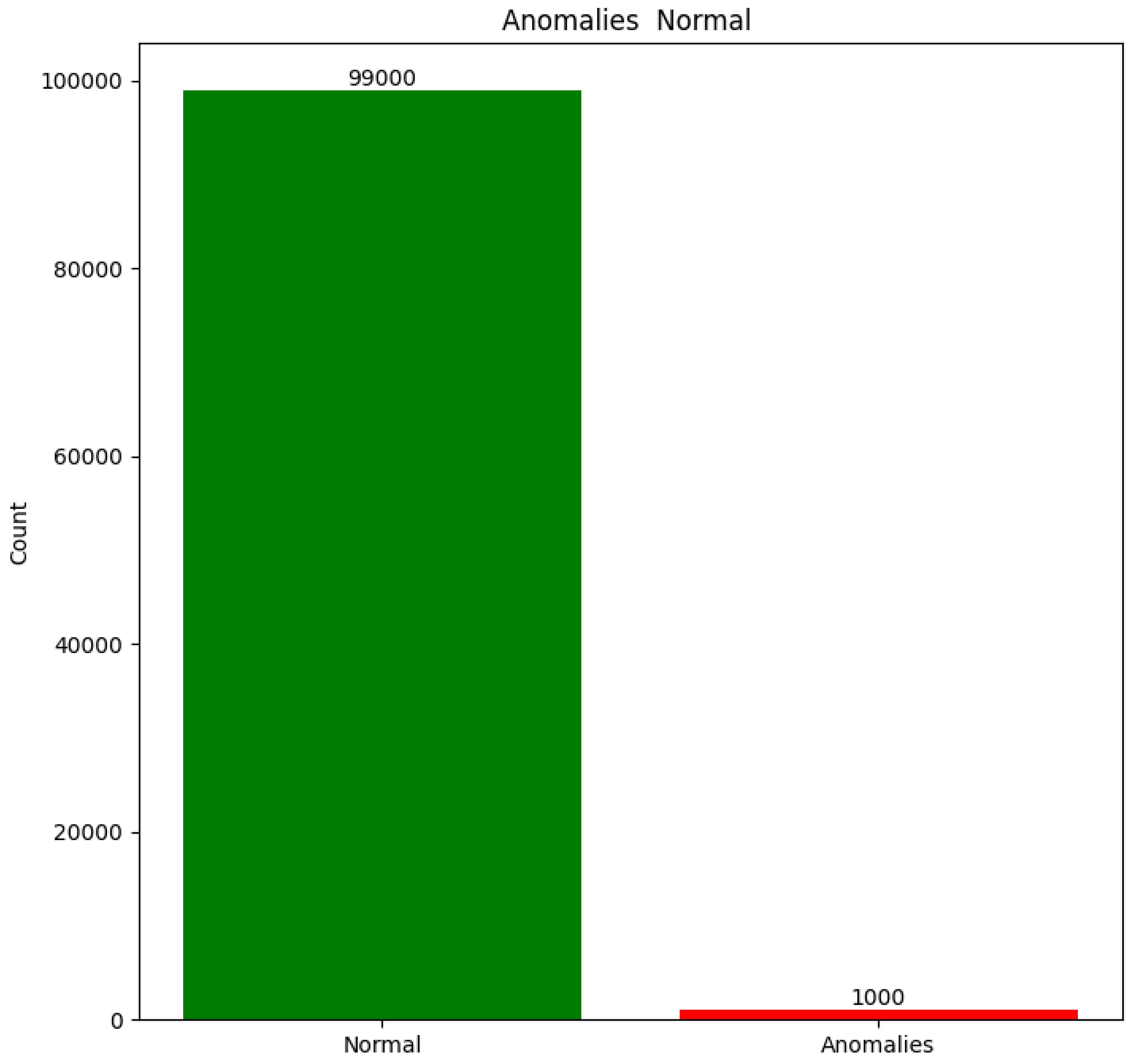
```
#separating normal and anomalies based on reconstruction_error
normal_errors = reconstruction_errors[~anomalies]
anomaly_errors = reconstruction_errors[anomalies]

#plotting histogram
fig, ax = plt.subplots(figsize=(6,6))
ax.hist(normal_errors ,bins=100 ,color='green' ,alpha=.5 ,label='Normal')
ax.hist(anomaly_errors ,bins=20 ,color='red' ,alpha=.5 ,label='Anomalies')
plt.axvline(x='threshold' ,color='k' ,linestyle='--', linewidth=1.3, label='Threshold')

plt.title('Reconstruction error histogram')
plt.xlabel('Reconstruction Error')
plt.ylabel('Frequency')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```




In [9]:

```
normal_errors = sum(~anomalies)
anomaly_errors = sum(anomalies)

plt.figure(figsize=(8, 8))
plot =  plt.bar(['Normal', 'Anomalies'], [normal_errors , anomaly_errors ], color=['green', 'red'])
plt.ylabel('Count')
plt.title('Anomalies  Normal ')
```

```
for bar in plot:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2.0, height, '%d' % int(height), ha='center', va='bottom')

plt.show()
```

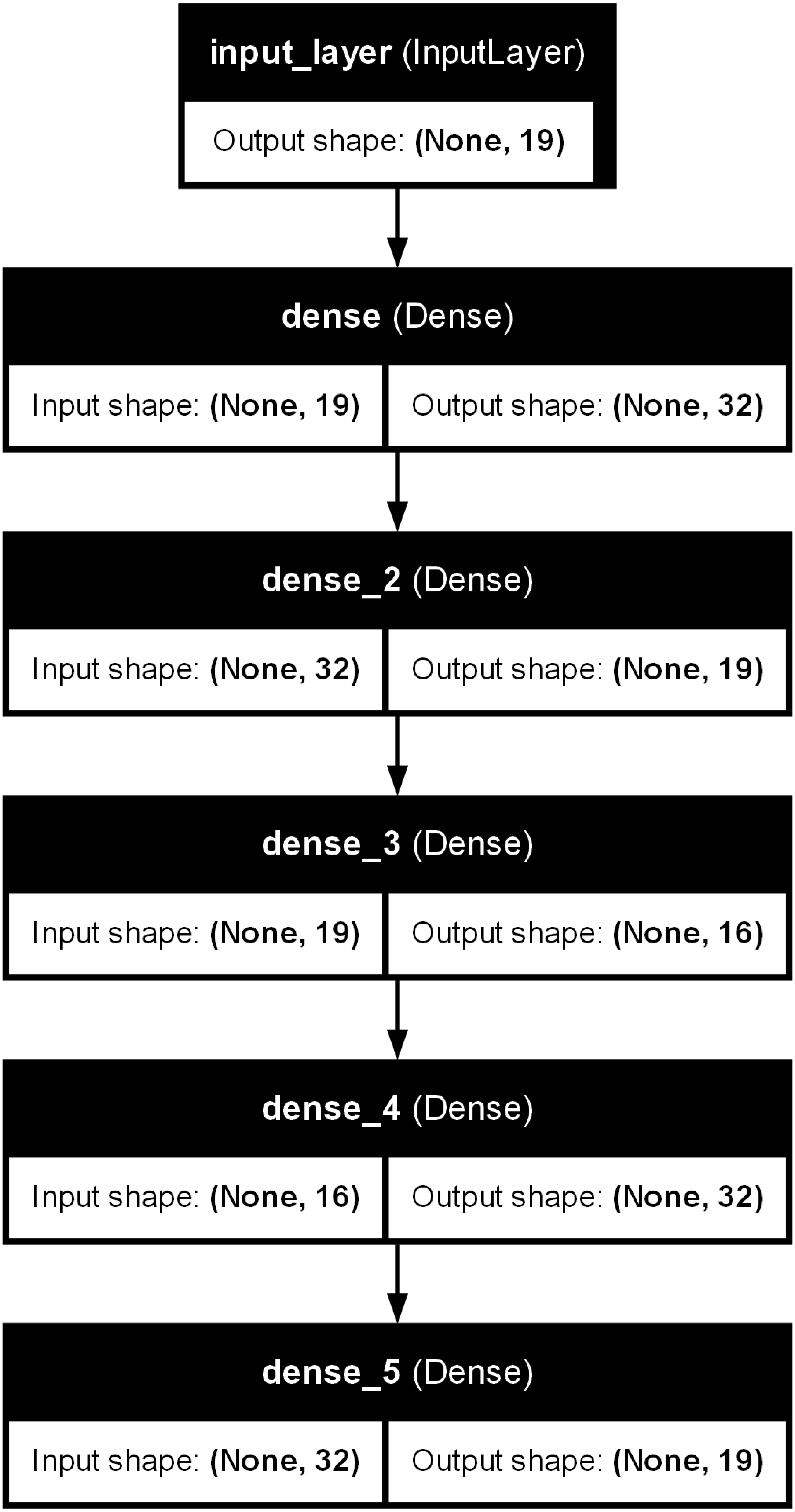



## Inferences from the Histogram :

- There are total 1000 anomalies are present in data based on the reconstruction error which cross the threshold .

In [10]:
```
plot_model(autoencoder,to_file='C:/Users/tmbha/Downloads/ifosys_springboard/autoencoder_model.png',show_shapes=
```

Out[10]:


input_layer (InputLayer)
Output shape: (None, 19)
dense (Dense)
Input shape: (None, 19)
Output shape: (None, 32)
dense_2 (Dense)
Input shape: (None, 32)
Output shape: (None, 19)
dense_3 (Dense)
Input shape: (None, 19)
Output shape: (None, 16)
dense_4 (Dense)
Input shape: (None, 16)
Output shape: (None, 32)
dense_5 (Dense)
Input shape: (None, 32)
Output shape: (None, 19)

## Inferences from the Architecture :

- There are total 5 layers are present 1 input and 4 Dense layer .

## Inferences from the Architecture :

- The image visually represent the model architecture representing layer names and their respective input and output .

In [11]:
```
df = pd.read_csv(r'C:\Users\tmbha\Downloads\ifosys_springboard\df_processed.csv')
```

In [12]:
```
df['Is_Anomaly'] = [1 if x == True else 0 for x in anomalies]
```

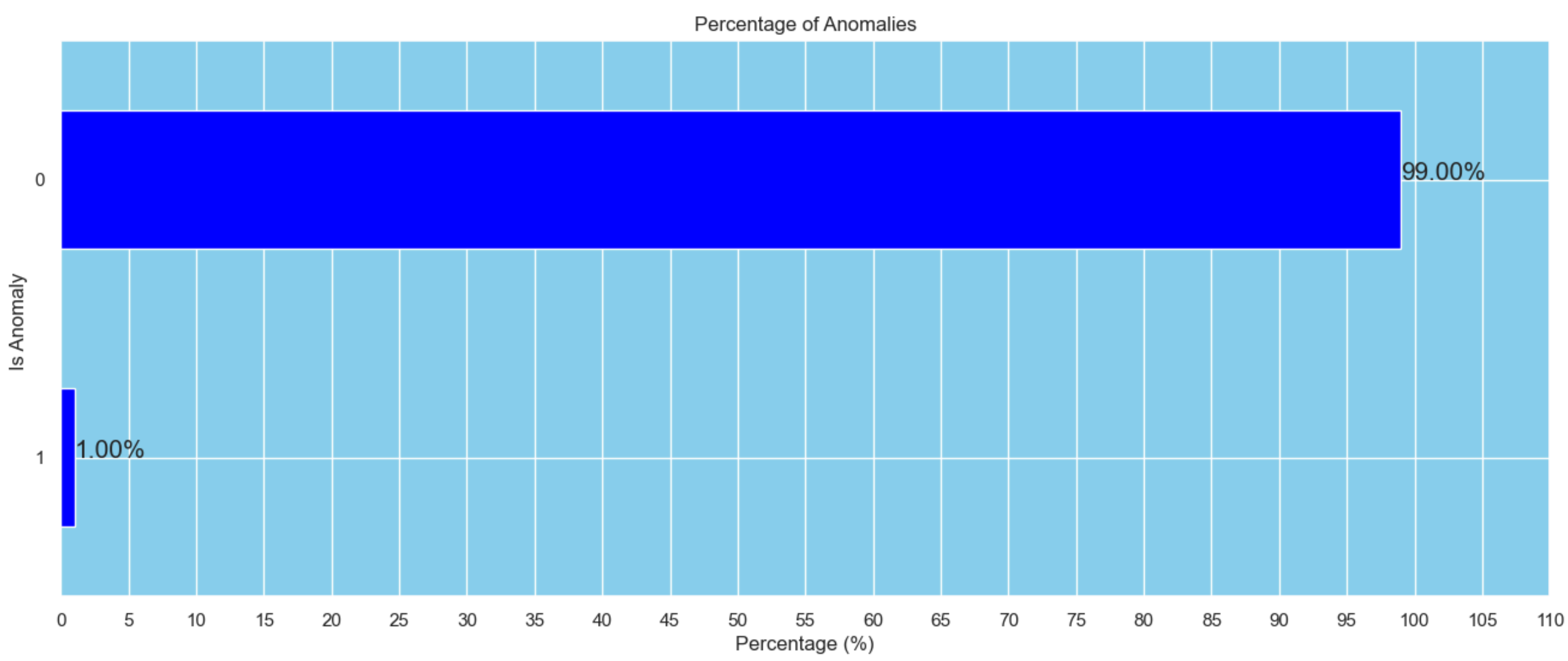
In [13]:
```
df.head(5).T
```

Out[13]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **Credentials of the Provider** | MD | MD | DPM | MD | DO |
| **Gender of the Provider** | F | F | M | M | M |
| **Entity Type of the Provider** | I | I | I | I | I |
| **City of the Provider** | SAINT LOUIS | FAYETTEVILLE | NORTH HAVEN | KANSAS CITY | JUPITER |
| **State Code of the Provider** | MO | NC | CT | MO | FL |
| **Country Code of the Provider** | US | US | US | US | US |
| **ProviderType** | Internal Medicine | Obstetrics & Gynecology | Podiatry | Internal Medicine | Internal Medicine |
| **Medicare Participation Indicator** | Y | Y | Y | Y | Y |
| **Place of Service** | F | O | O | O | O |
| **HCPCS Description** | Initial hospital inpatient care, typically 70 ... | Screening mammography, bilateral (2-view study... | Established patient home visit, typically 25 m... | Urinalysis, manual test | Injection beneath the skin or into muscle for ... |
| **HCPCS Drug Indicator** | N | N | N | N | N |
| **Number of Services** | 27.0 | 175.0 | 32.0 | 20.0 | 33.0 |
| **Number of Medicare Beneficiaries** | 24 | 175 | 13 | 18 | 24 |
| **Number of Distinct Medicare Beneficiary/Per Day Services** | 27 | 175 | 32 | 20 | 31 |
| **Average Medicare Allowed Amount** | 200.587778 | 123.73 | 90.65 | 3.5 | 26.52 |
| **Average Submitted Charge Amount** | 305.211111 | 548.8 | 155.0 | 5.0 | 40.0 |
| **Average Medicare Payment Amount** | 157.262222 | 118.83 | 64.439688 | 3.43 | 19.539394 |
| **Average Medicare Standardized Amount** | 160.908889 | 135.315257 | 60.595937 | 3.43 | 19.057576 |
| **Name** | UPADHYAYULA SATYASREE | JONES WENDY P | DUROCHER RICHARD W | FULLARD JASPER | PERROTTI ANTHONY E |
| **Full Address** | 1402 S GRAND BLVD FDT 14TH FLOOR | 2950 VILLAGE DR | 20 WASHINGTON AVE STE 212 | 5746 N BROADWAY ST | 875 MILITARY TRL SUITE 200 |
| **Is_Anomaly** | 0 | 0 | 0 | 0 | 0 |

In [14]:
```
# Set seaborn plot style
sns.set(rc={'axes.facecolor': 'skyblue'}, style='darkgrid')

# Calculate the percentage of anomalies
```

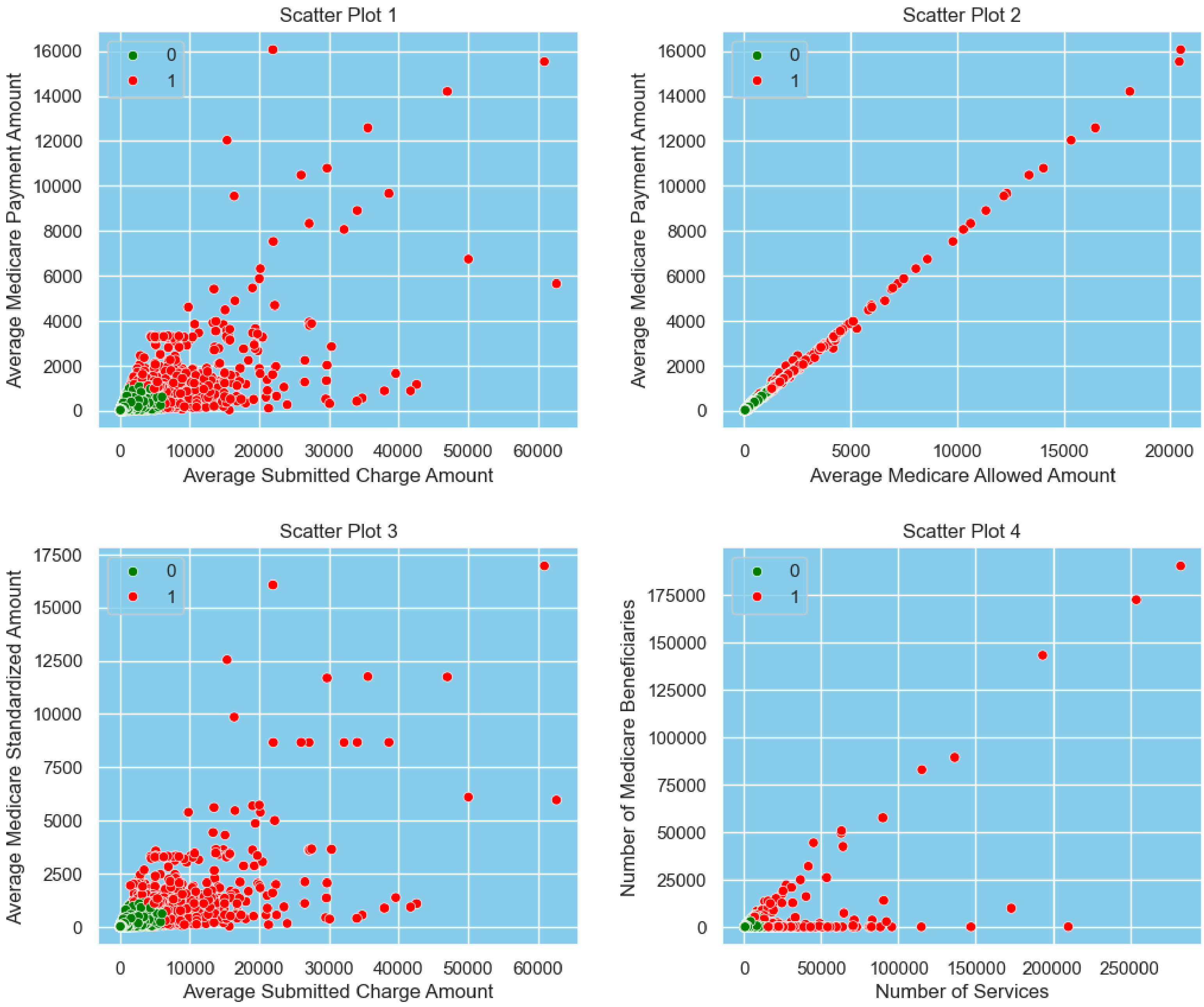
```
anomalies_percentage = df['Is_Anomaly'].value_counts(normalize=True) * 100

# Plotting the percentage of anomalies
plt.figure(figsize=(16,6))
anomalies_percentage.plot(kind='barh', color='Blue')

# Adding the percentage labels on the bars
for index, value in enumerate(anomalies_percentage):
    plt.text(value, index, f'{value:.2f}%', fontsize=15)

plt.title('Percentage of Anomalies')
plt.xticks(ticks=np.arange(0, 115, 5))
plt.xlabel('Percentage (%)')
plt.ylabel('Is Anomaly ')
plt.gca().invert_yaxis()
plt.show()
```




## Inferences from the Graph :

- There are total **1.00%** anomalies are detected by our model i.e.around 1000 entries in entire dataset .

In [15]:
```
sns.set(rc={'axes.facecolor': 'skyblue'}, style='darkgrid')
fig, axs = plt.subplots(2,2, figsize=(12,10))

#Scatter plot 1
sns.scatterplot(x='Average Submitted Charge Amount', y='Average Medicare Payment Amount',data=df, hue='Is_A
                palette=['green','red'])
axs[0,0].set_title('Scatter Plot 1')
axs[0,0].set_xlabel('Average Submitted Charge Amount')
axs[0,0].set_ylabel('Average Medicare Payment Amount')
axs[0,0].legend()

#Scatter plot 2
sns.scatterplot(x='Average Medicare Allowed Amount', y='Average Medicare Payment Amount',data=df, hue='Is_A
                palette=['green','red'])
axs[0,1].set_title('Scatter Plot 2')
axs[0,1].set_xlabel('Average Medicare Allowed Amount')
axs[0,1].set_ylabel('Average Medicare Payment Amount')
axs[0,1].legend()

#Scatter plot 3
sns.scatterplot(x='Average Submitted Charge Amount', y='Average Medicare Standardized Amount',data=df, hue=
                palette=['green','red'])
axs[1,0].set_title('Scatter Plot 3')
axs[1,0].set_xlabel('Average Submitted Charge Amount')
axs[1,0].set_ylabel('Average Medicare Standardized Amount')
axs[1,0].legend()

#Scatter plot 4
sns.scatterplot(x='Number of Services', y='Number of Medicare Beneficiaries',data=df, hue='Is_Anomaly',ax=a
                palette=['green','red'])
axs[1,1].set_title('Scatter Plot 4')
axs[1,1].set_xlabel('Number of Services')
axs[1,1].set_ylabel('Number of Medicare Beneficiaries')
axs[1,1].legend()

plt.subplots_adjust(wspace=0.3,hspace=0.3)

plt.show()
```

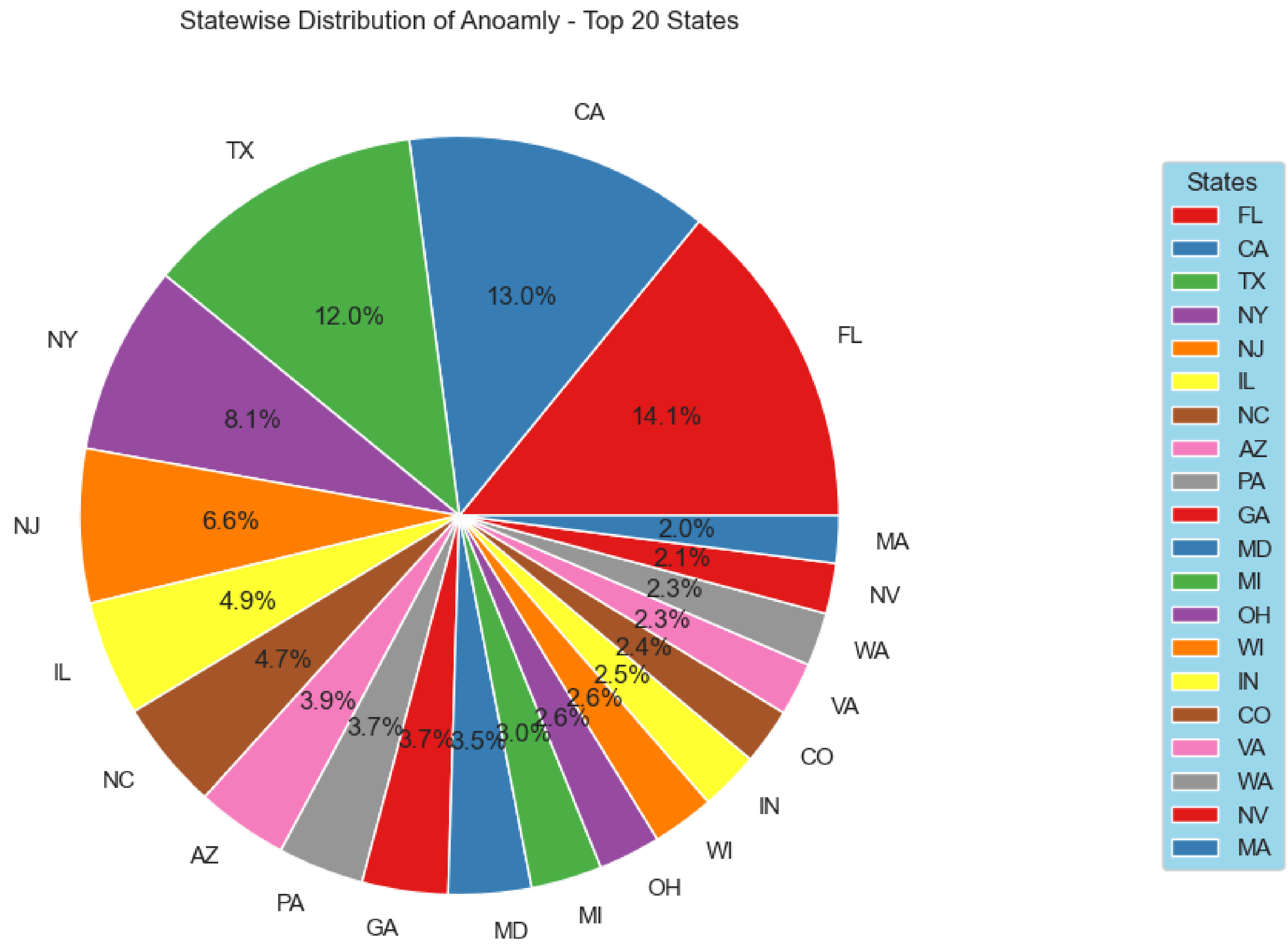



## Inferences from the Scatter-Plots :

- By Visualizing the **Scatter Plots** we can see that our **Autoencoder** works well, and our model is able to distinguish between Normal points and Anomalies.
- There is **clear separation** can be seen between the normal and anomalous point
- **Green** dots indicates the **Normal** points while **red** dots indicates **Anomaly**.

In [16]:

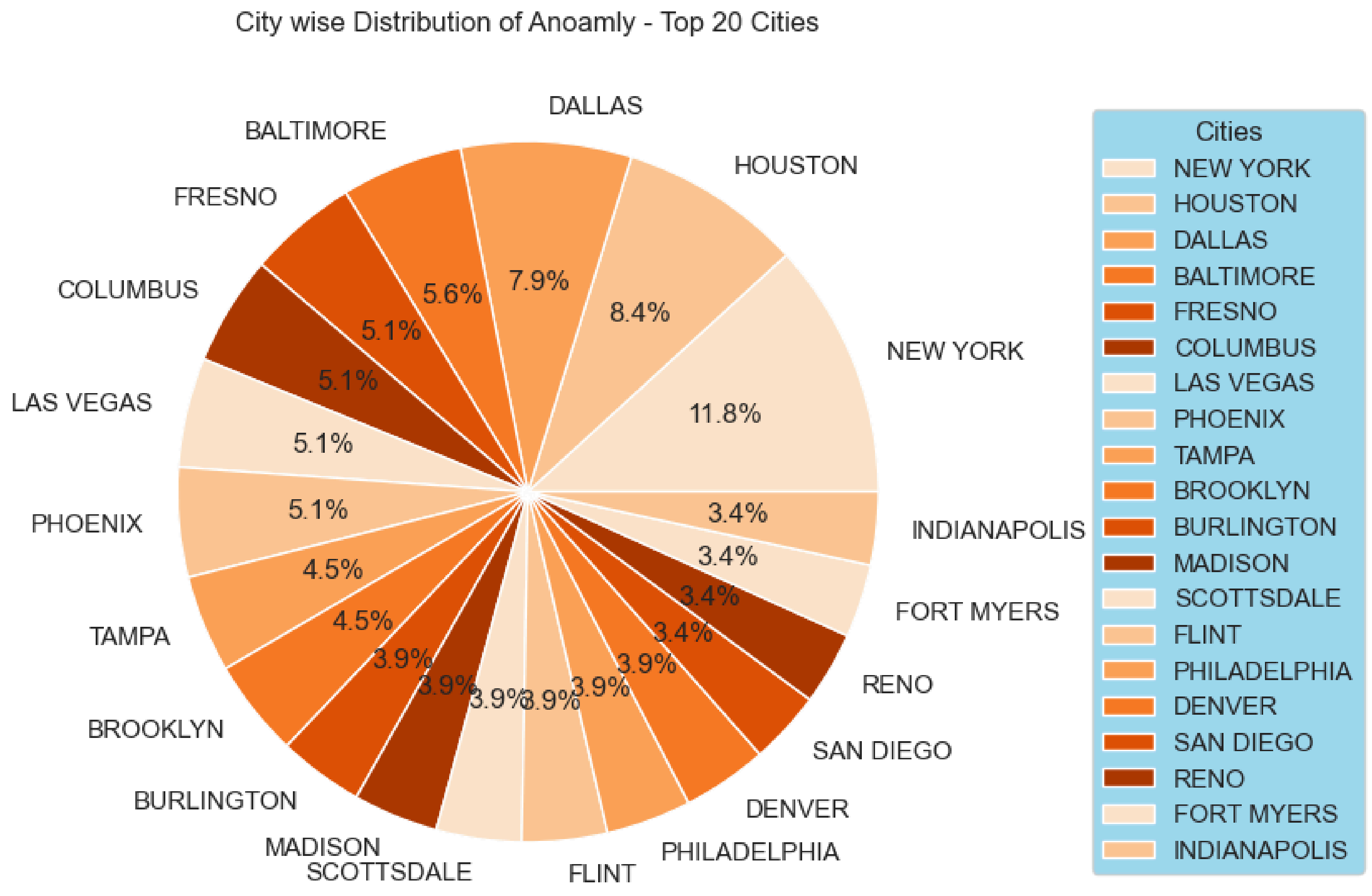
```
#filtering States with rows which has anomaly
States_with_anomalies = df[df['Is_Anomaly']==1]['State Code of the Provider']

# counting the States with occurence
State_counts = States_with_anomalies.value_counts(normalize=True).head(20)

#creating pie chart
plt.figure(figsize=(7,8))
plt.pie(State_counts,labels=State_counts.index, autopct='%1.1f%%',colors=sns.color_palette('Set1'))
plt.axis('equal')
plt.title('Statewise Distribution of Anoamly - Top 20 States')
plt.legend(title='States' ,loc='center right',bbox_to_anchor=(1,0,0.5,1))
plt.show()
```




## Inferences from the Pie-Chart :

- The pie chart shows **Top 20 state** with anomaly percentage where **Florida** has highest no of anomalies **14.1%** followed by **California 13.0%**, **Texas 12.0%** .

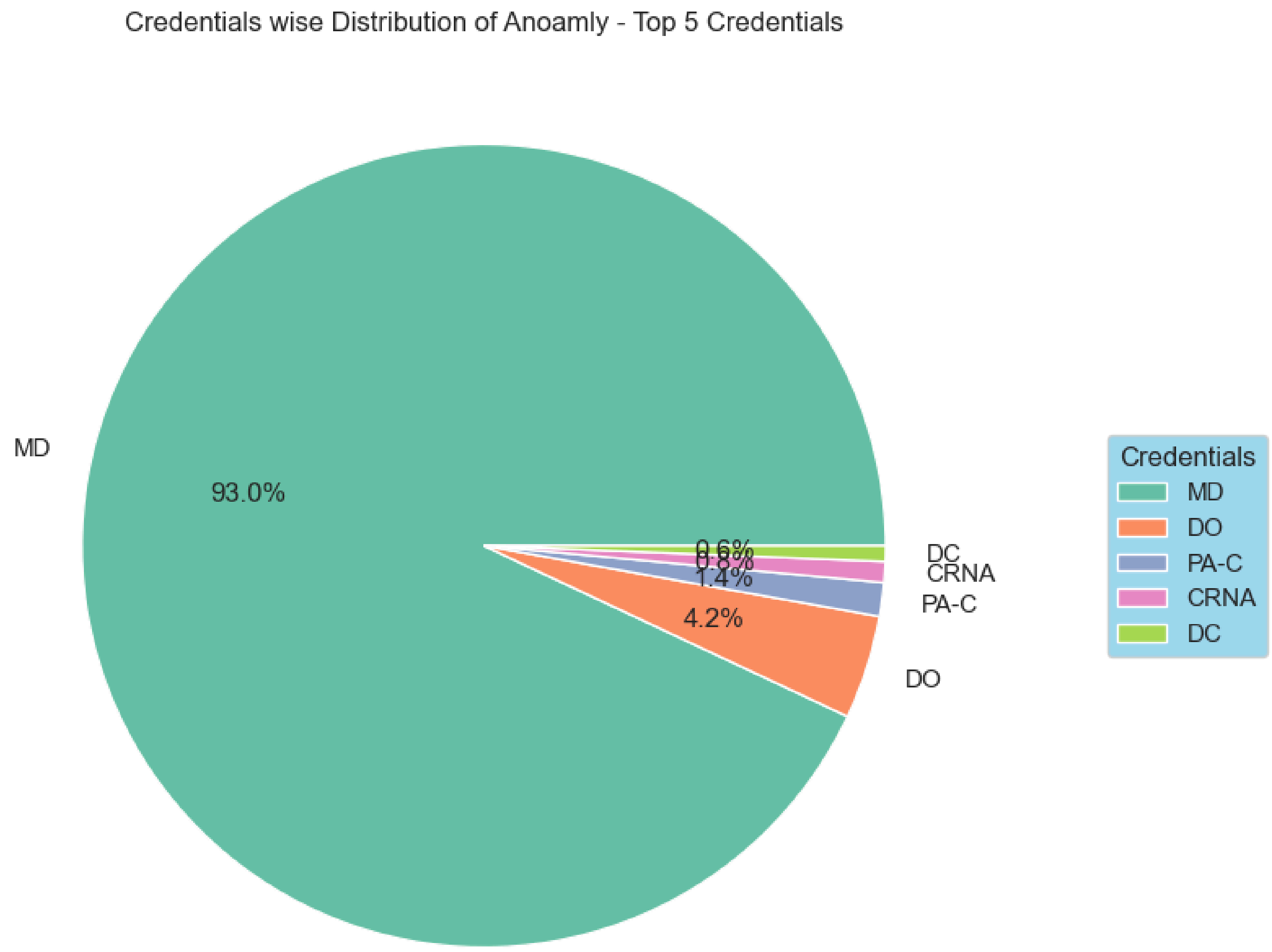
In [17]:
```
#filtering cities with rows which has anomaly
cities_with_anomalies = df[df['Is_Anomaly']==1]['City of the Provider']

# counting the cities with occurence
city_counts = cities_with_anomalies.value_counts(normalize=True).head(20)

#creating pie chart
plt.figure(figsize=(6,7))
plt.pie(city_counts,labels=city_counts.index, autopct='%1.1f%%',colors=sns.color_palette('Oranges'))
plt.title('City wise Distribution of Anoamly - Top 20 Cities')
plt.legend(title='Cities' ,loc='center right',bbox_to_anchor=(1,0,0.6,1))
plt.axis('equal')
plt.show()
```




## Inferences from the Pie-Chart :

- The pie chart shows **Top 20 Cities** with anomaly percentage where **New York** has highest no of anomalies **11.8%** followed by **Houston 8.4%** and **Dallas 7.9%** .

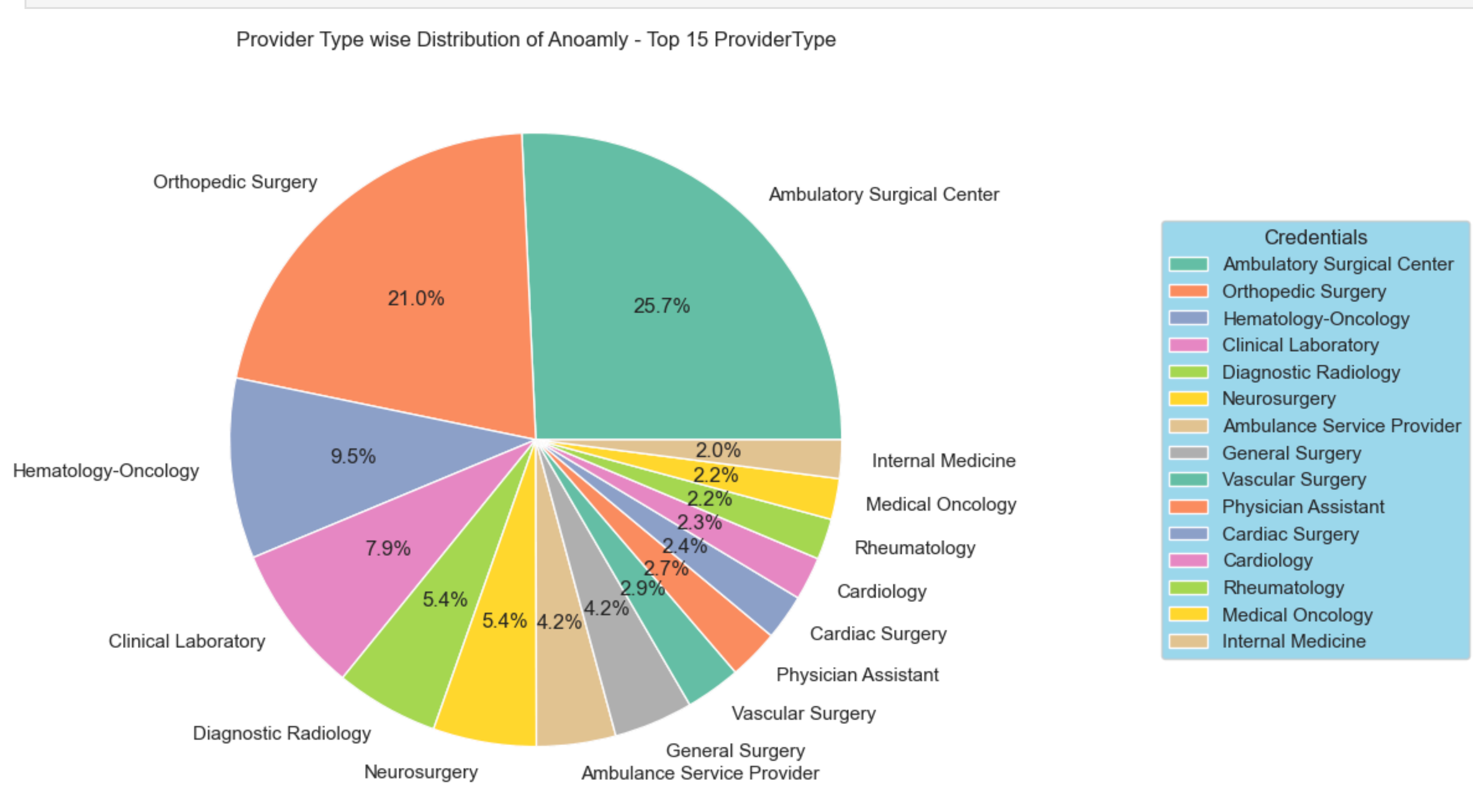
In [18]:
```
#filtering Credentials of the Provider with rows which has anomaly
Credentials_with_anomalies = df[df['Is_Anomaly']==1]['Credentials of the Provider']

# counting the Credentials with occurence
Credential_counts = Credentials_with_anomalies.value_counts(normalize=True).head(5)

#creating pie chart
plt.figure(figsize=(7,8))
plt.pie(Credential_counts,labels=Credential_counts.index, autopct='%1.1f%%',colors=sns.color_palette('S
plt.title('Credentials wise Distribution of Anoamly - Top 5 Credentials')
plt.axis('equal')
plt.legend(title='Credentials' ,loc='center right',bbox_to_anchor=(1,0,0.4,1))
plt.show()
```




## Inferences from the Pie-Chart :

- The pie chart shows **Top 5 Credentials** with anomaly percentage where **MD** has highest no of anomalies **93.0%** followed by **DO 4.2%**, **PA-C 1.4%** .

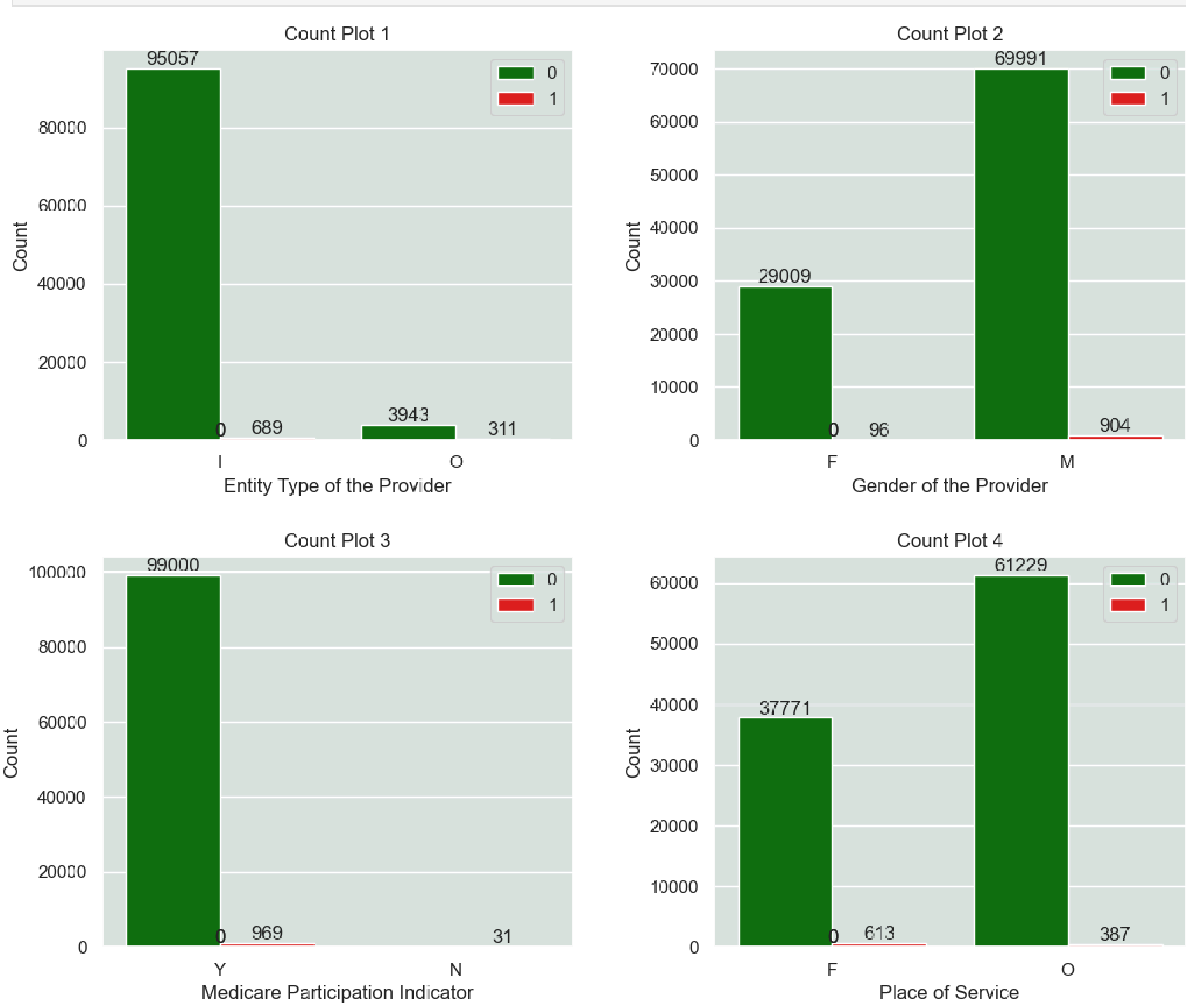
In [19]:
```
#filtering ProviderType with rows which has anomaly
ProviderType_with_anomalies = df[df['Is_Anomaly']==1]['ProviderType']

# counting the ProviderType with occurence
ProviderType_counts = ProviderType_with_anomalies.value_counts(normalize=True).head(15)

#creating pie chart
plt.figure(figsize=(7,8))
plt.pie(ProviderType_counts,labels=ProviderType_counts.index, autopct='%1.1f%%',colors=sns.color_pale
plt.title('Provider Type wise Distribution of Anoamly - Top 15 ProviderType')
plt.axis('equal')
plt.legend(title='Credentials' ,loc='center right',bbox_to_anchor=(1,0,0.9,1))
plt.show()
```




## Inferences from the Pie-Chart :

- The pie chart shows **Top 15 ProviderType** with anomaly percentage where **Ambulatory Surgical Center** has highest no of anomalies **25.7%** followed by **Orthopedic Surgery 21.0%**, **Hematology-Oncology 9.5%** .

In [20]:
```
sns.set(rc={'axes.facecolor': '#DAE5E0'}, style='darkgrid')
fig, axs = plt.subplots(2,2, figsize=(12,10))

#Count plot 1
sns.countplot(x='Entity Type of the Provider',data=df, hue='Is_Anomaly',ax=axs[0,0],palette=['green'
axs[0,0].set_title('Count Plot 1')
axs[0,0].set_xlabel('Entity Type of the Provider')
axs[0,0].set_ylabel('Count')
axs[0,0].legend()

#Count plot 2
sns.countplot(x='Gender of the Provider',data=df, hue='Is_Anomaly',ax=axs[0,1],palette=['green','rec
axs[0,1].set_title('Count Plot 2')
axs[0,1].set_xlabel('Gender of the Provider')
axs[0,1].set_ylabel('Count')
axs[0,1].legend()

#Count plot 3
sns.countplot(x='Medicare Participation Indicator',data=df, hue='Is_Anomaly',ax=axs[1,0],palette=['g
axs[1,0].set_title('Count Plot 3')
axs[1,0].set_xlabel('Medicare Participation Indicator')
axs[1,0].set_ylabel('Count')
axs[1,0].legend()

#Count plot 1
sns.countplot(x='Place of Service',data=df, hue='Is_Anomaly',ax=axs[1,1],palette=['green','red'])
axs[1,1].set_title('Count Plot 4')
axs[1,1].set_xlabel('Place of Service')
axs[1,1].set_ylabel('Count')
axs[1,1].legend()

for ax in axs.flat:
    for p in ax.patches:
        ax.text(p.get_x() + p.get_width()/2, p.get_height(), '%d' % int(p.get_height()), ha='center'

plt.subplots_adjust(wspace=0.3,hspace=0.3)

plt.show()
```




## Inferences from the Count Plots :

- The Countplots shows the anomaly in the **categorical columns** **red** bars shows **anomaly** and the **green** bars represent the **normal** point.
- **Count Plot 1** shows anomaly in the **Entity** where **I-individual** has **689** **anomalies** where **O-organization** has **311** **anomalies** only which indicates that Individual entity has more fraudulent transactions.
- **Count Plot 2** shows anomaly in the **Gender** where **F-Female** has **96** **anomalies** where **M-Male** has **904** **anomalies** which indicates that in Male has more fraudulent transactions.
- **Count Plot 3** shows anomaly in the **Medicare Participation Indicator** where **Y-Yes** has **969** **anomalies** where **N-No** has **31** **anomalies** only which indicates that Yes has major fraudulent transactions.
- **Count Plot 4** shows anomaly in the **Place of Service** where **F-Facility** has **613** **anomalies** where **O-Non-Facility** has **387** **anomalies** which indicates that Facility Places has more no of fraudulent transactions.