INFOSYS SPRINGBOARD

Building Anomaly Detection System using Python

(unsupervised 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]:
```
#importing necessary libraries
#Loading dataset

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import statistics
from matplotlib.colors import LinearSegmentedColormap
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
import category_encoders as ce
from sklearn.decomposition import PCA
from sklearn.covariance import EllipticEnvelope
from sklearn.svm import OneClassSVM
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import shap

df = pd.read_csv(r'C:\Users\tmbha\Downloads\ifosys_springboard\df_processed.csv')  #orignal Dataframe after EDA
hdata_scaled = pd.read_csv(r'C:\Users\tmbha\Downloads\ifosys_springboard\test.csv') #scaled Dataframe after scaling milestone 2
```

```
C:\Users\tmbha\AppData\Local\Programs\Python\Python310\lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Plea
se update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```
#overview of orignal dataframe after EDA
df_i = df #making copy of dataset for further use in isolation approach
df_s = df #making copy of dataset for further use in one SVM approach
```

In [3]:
```
#overview of scaled dataframe
hdata_scaled.sample(5)
```

Out[3]:

	Number of Services	Number of Medicare Beneficiaries	Number of Distinct Medicare Beneficiary/Per Day Services	Average Medicare Allowed Amount	Average Submitted Charge Amount	Average Medicare Payment Amount	Average Medicare Standardized Amount	Diff_submitted_allowed	Is_US	Gender	Ent
90326	-0.070862	-0.053000	-0.048235	-0.234970	-0.104037	-0.237869	-0.235825	-0.056343	1	1	
12579	-0.065648	-0.016050	-0.040309	-0.267469	-0.277197	-0.280491	-0.267132	-0.253844	1	1	
86786	-0.065648	-0.030470	-0.040309	-0.125773	-0.205625	-0.129553	-0.102566	-0.209312	1	0	
36405	0.161372	0.220069	0.304767	-0.102488	-0.192060	-0.128449	-0.123258	-0.199839	1	1	
38833	-0.074873	-0.048494	-0.054332	0.001655	-0.014634	0.003752	0.021090	-0.017965	1	0	

Step 10 | Model Building

Using Isolation Forest algorithm

- Given the multi-dimensional nature of the data, it would be prudent to use algorithms that can detect anomalies in multi-dimensional spaces. I am going to use the **Isolation Forest** algorithm for this task. This algorithm works well for multi-dimensional data and is computationally efficient. It isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature

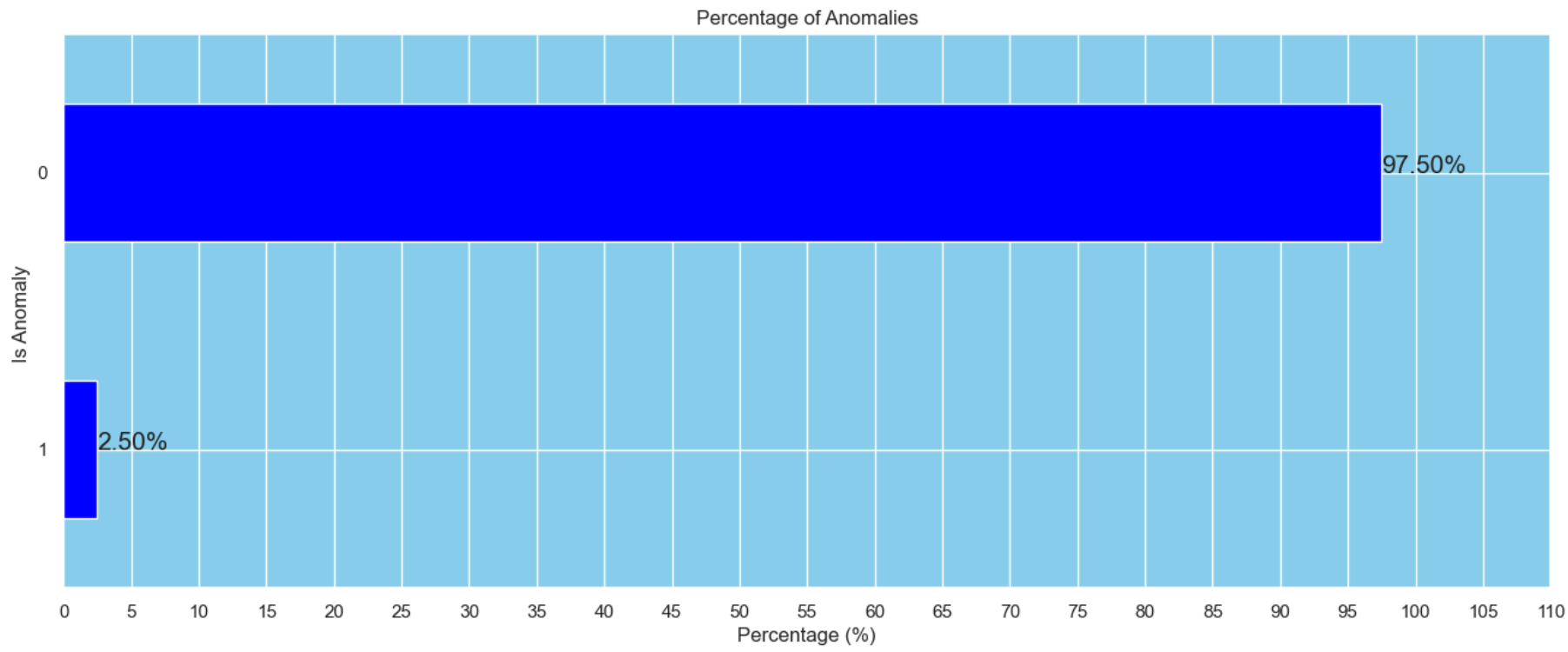
In [4]:
```
# Initializing the IsolationForest model with a contamination parameter of auto
#max_feature parameter 1 to 9
#using Default parameters as it's unsupervised approach considering we don't know the depth of data
model = IsolationForest(n_estimators=100, contamination=0.025, random_state=42, max_samples='auto')

# Fitting the model on our dataset (converting DataFrame to NumPy to avoid warning) on orignal dataframe
df_i['Anomaly_Scores'] = model.fit_predict(hdata_scaled.iloc[:, 1:].to_numpy())

# Creating a new column to identify anomalies (1 for inliers and -1 for outliers) on orignal dataframe
df_i['Is_Anomaly'] = [1 if x == -1 else 0 for x in df_i['Anomaly_Scores']]

# Display the first few rows of the orignal dataframe
df_i.sample(5).T
```

Out[4]:

	49894	71692	61404	57229	96476
Credentials of the Provider	DO	MD	DO	NP	MD
Gender of the Provider	F	F	F	F	M
Entity Type of the Provider	I	I	I	I	I
City of the Provider	BURLINGTON	MYRTLE BEACH	DALLAS	MCMINNVILLE	VENTURA
State Code of the Provider	VT	SC	TX	TN	CA
Country Code of the Provider	US	US	US	US	US
ProviderType	Surgical Oncology	Internal Medicine	Internal Medicine	Nurse Practitioner	Diagnostic Radiology
Medicare Participation Indicator	Y	Y	Y	Y	Y
Place of Service	F	O	F	O	F
HCPCS Description	Ultrasound of one breast	Electrocardiogram, routine ecg with 12 leads; ...	Subsequent hospital inpatient care, typically ...	Injection, triamcinolone acetonide, not otherw...	CT scan of lower spine
HCPCS Drug Indicator	N	N	N	Y	N
Number of Services	43.0	12.0	349.0	56.0	11.0
Number of Medicare Beneficiaries	39	12	162	13	11
Number of Distinct Medicare Beneficiary/Per Day Services	43	12	349	14	11
Average Medicare Allowed Amount	34.38	15.73	73.46	1.854286	52.647273
Average Submitted Charge Amount	38.0	65.0	202.11447	15.0	321.0
Average Medicare Payment Amount	22.802093	7.296667	56.020888	1.27125	37.421818
Average Medicare Standardized Amount	22.37186	8.7175	56.573754	1.511607	34.241818
Name	SOWDEN MICHELLE M	TANGEMAN LINDA	HUYNH SUSAN	JENKINS ELIZABETH A	MCMAHAN JOHN
Full Address	89 BEAUMONT AVE GIVEN BUILDING - B227	141 MCDONALD CT	1441 N BECKLEY AVE	155 HEALTH WAY SUITE 1	3291 LOMA VISTA RD
Anomaly_Scores	1	1	1	1	1
Is_Anomaly	0	0	0	0	0

In [5]:
```
# Set seaborn plot style
sns.set(rc={'axes.facecolor': 'skyblue'}, style='darkgrid')
```

```
# Calculate the percentage of anomalies
anomalies_percentage = df_i['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()
```

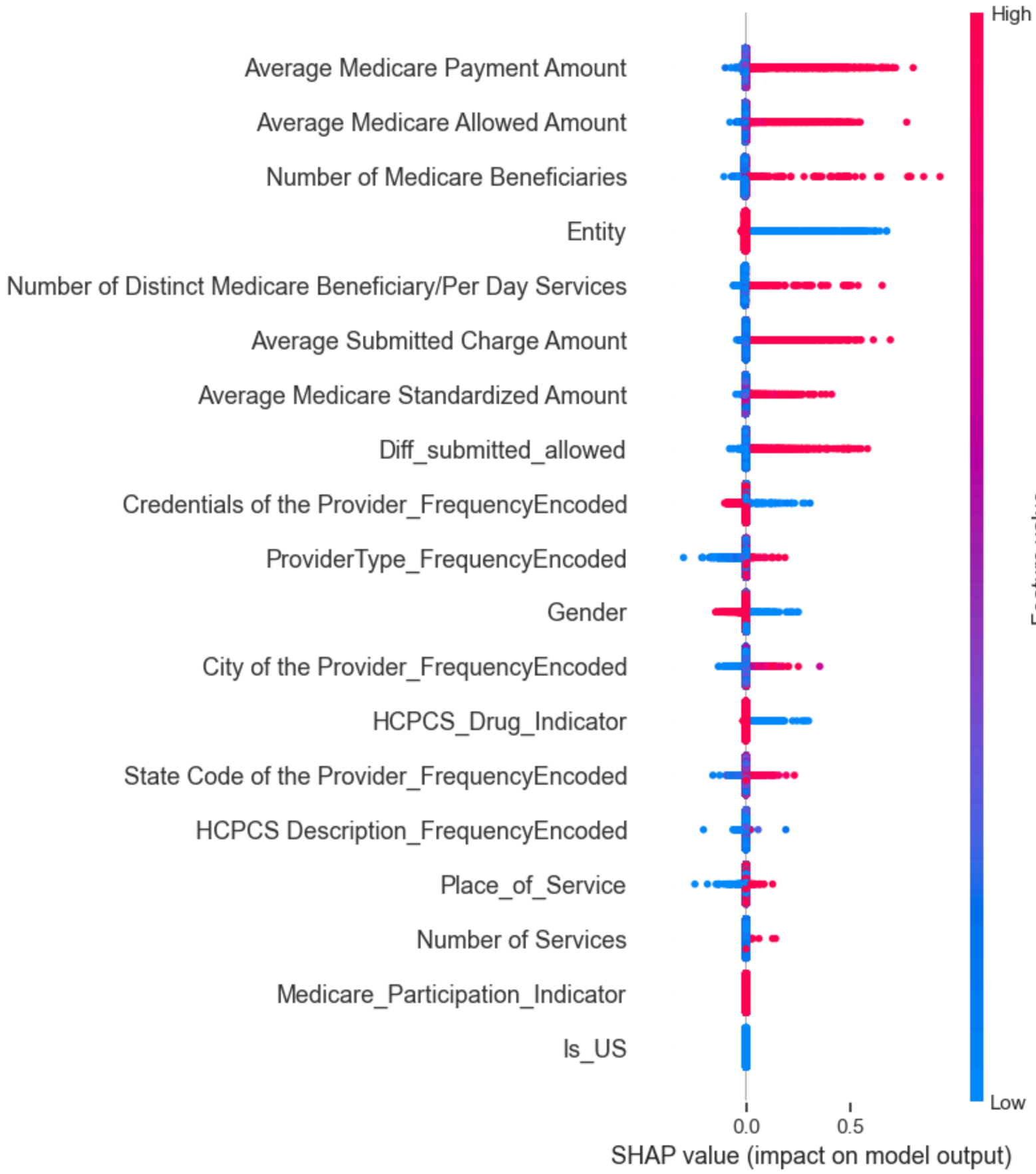


Inferences from the Graph :

- There are total **2.50%** anomalies are detected by our model i.e.around 2500 entries in entire dataset .

In [15]:
```
sns.set(rc={'axes.facecolor': 'white'}, style='darkgrid')

#creating new dataframe with scaled data and anomaly column as target variable for shap analysis
iso = hdata_scaled.join(df_i['Is_Anomaly'], how='inner')

#creating sample size due to system constraints
#iso_sample = iso.sample(n=10000)

#splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(iso.drop('Is_Anomaly',axis=1),iso['Is_Anomaly'],test_size=0.2, random_s

#Reseting Index of X_test
X_test.reset_index(drop=True,inplace=True)


#training logistic regression model on data
model = LogisticRegression()
model.fit(X_train, y_train)

#creating a SHAP explainer object
explainer = shap.Explainer(model.predict,X_train,algorithm='permutation')

#computing shap values for the testing data
shap_values = explainer.shap_values(X_test)

#Plotting the SHAP values using a summary plot
shap.summary_plot(shap_values, X_test, show=False)



plt.show()
```

```
PermutationExplainer explainer: 20001it [07:42, 41.75it/s]
```

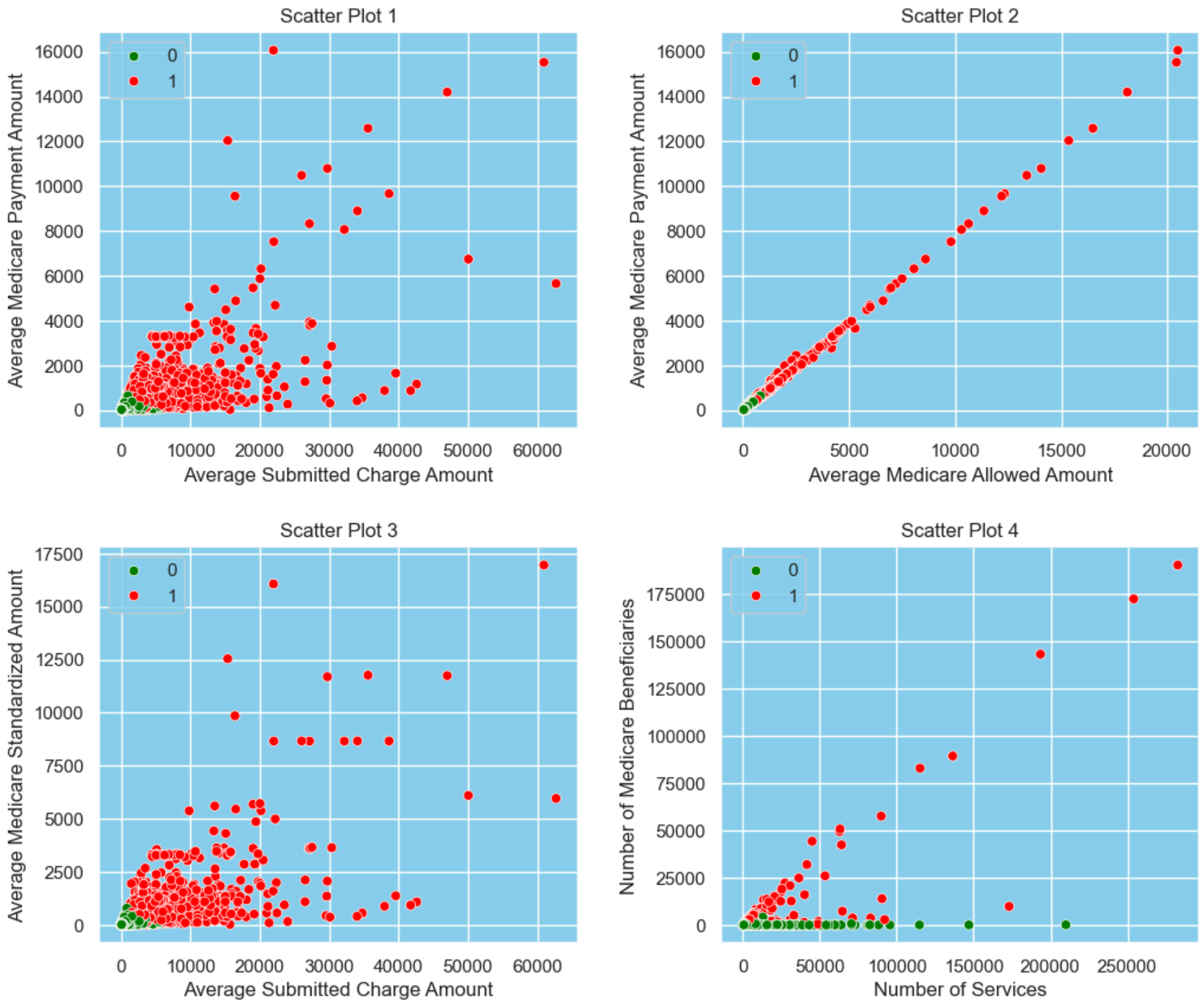


Inferences from the Summary Plot of Isolation Forest :

- There is major **positive** infulence of columns on model such columns are **Average Medicare Payment Amount**,**Average Medicare Allowed Amount** overall all numerical columns are has impact on model either positive or negative some categorical columns such as **city** and **State** of the provider also has some contributions to model .

In [26]:
```
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_i, hue='Is_Anomaly',ax=ax
                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_i, hue='Is_Anomaly',ax=ax
                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_i, hue='Is_Anomaly',
                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_i, hue='Is_Anomaly',ax=axs[1,1],
                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()
```

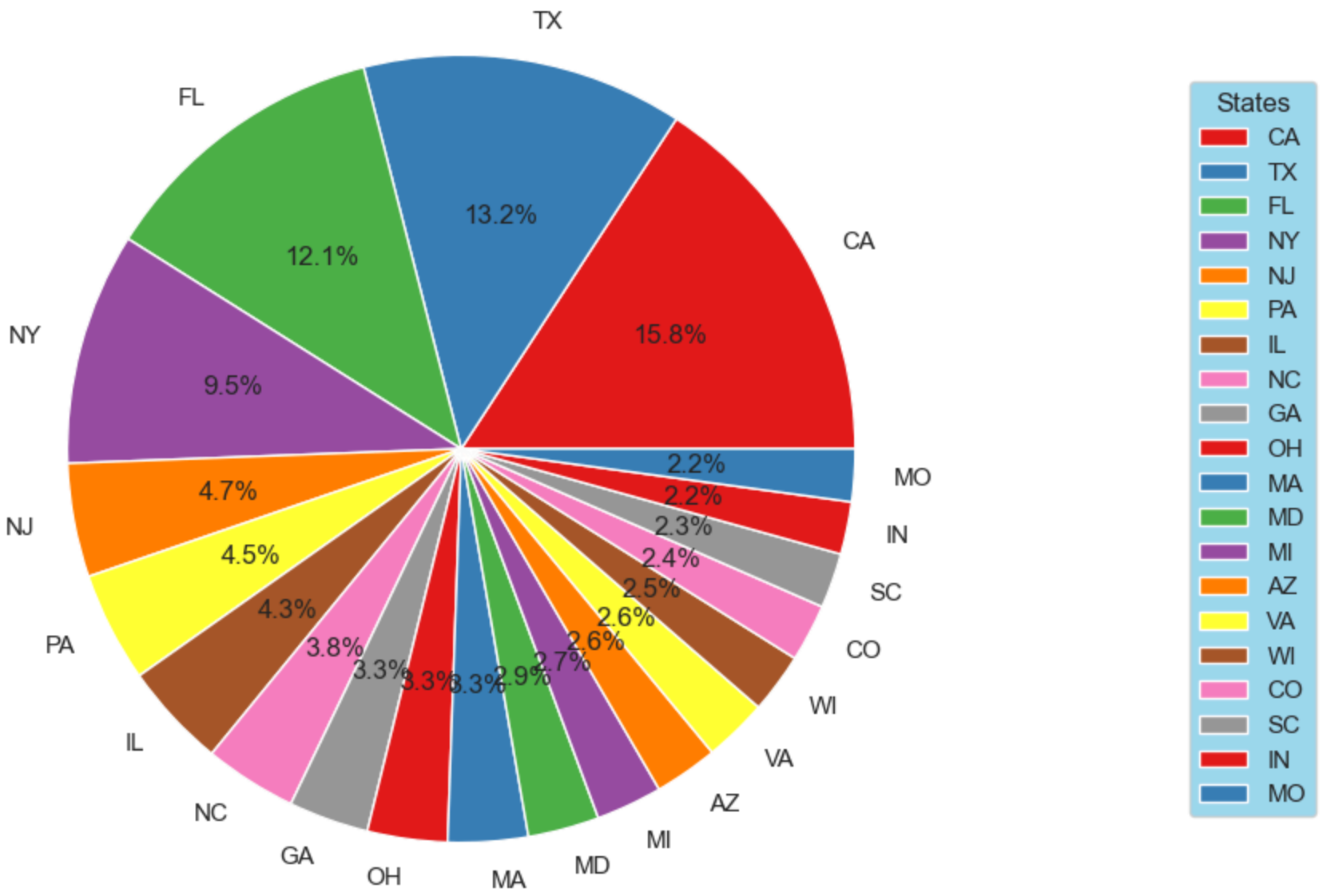


Inferences from the Scatter-Plots :

- By Visualizing the **Scatter Plots** we can see that our **Isolation Forest** algorithm 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 [27]:

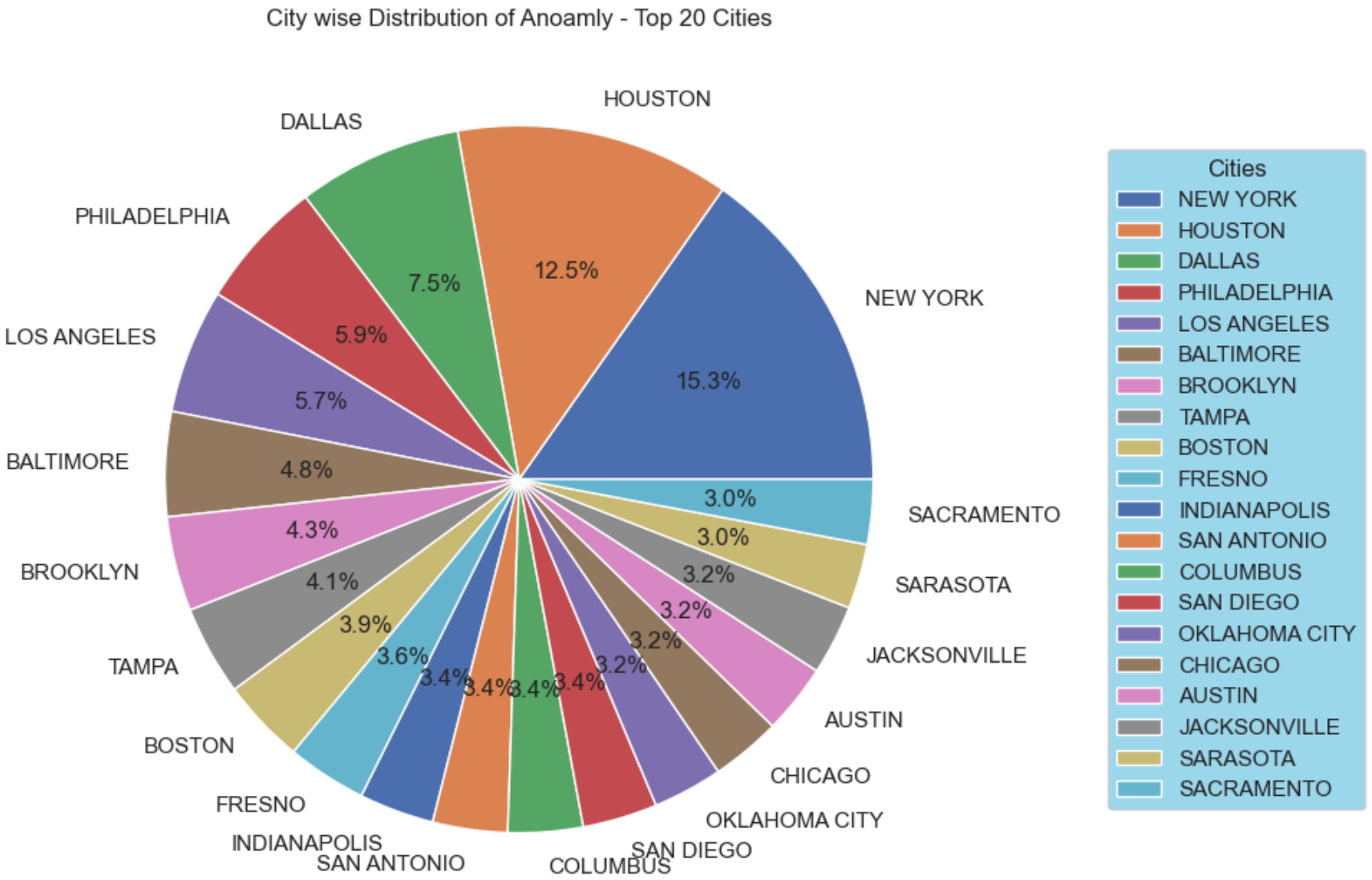
```
#filtering States with rows which has anomaly
States_with_anomalies = df_i[df_i['Is_Anomaly']==1]['State Code of the Provider']

# 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 **California** has highest no of anomalies **15.8%** followed by **Texas 13.2%**, **Florida 12.2%** .

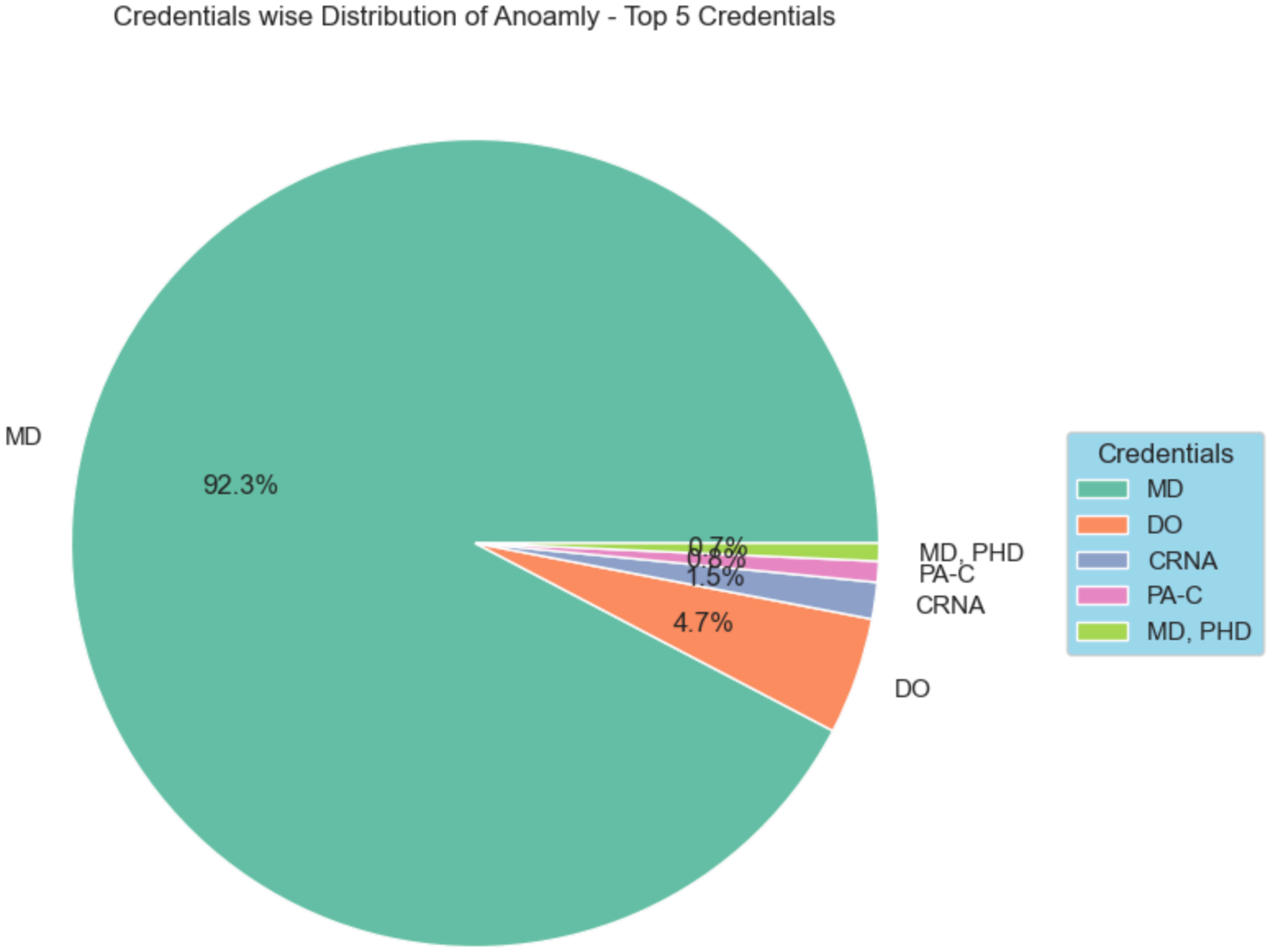
In [28]:
```
#filtering cities with rows which has anomaly
cities_with_anomalies = df_i[df_i['Is_Anomaly']==1]['City of the Provider']

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

#creating pie chart
plt.figure(figsize=(7,8))
plt.pie(city_counts,labels=city_counts.index, autopct='%1.1f%%')
plt.title('City wise Distribution of Anoamly - Top 20 Cities')
plt.axis('equal')
plt.legend(title='Cities' ,loc='center right',bbox_to_anchor=(1,0,0.6,1))
plt.show()
```



Inferences from the Pie-Chart :

- The pie chart shows **Top 20 Cities** with anomaly percentage where **NEWYORK** has highest no of anomalies **15.3%** followed by **Houston 12.5%** and **Dallas 7.5%** .

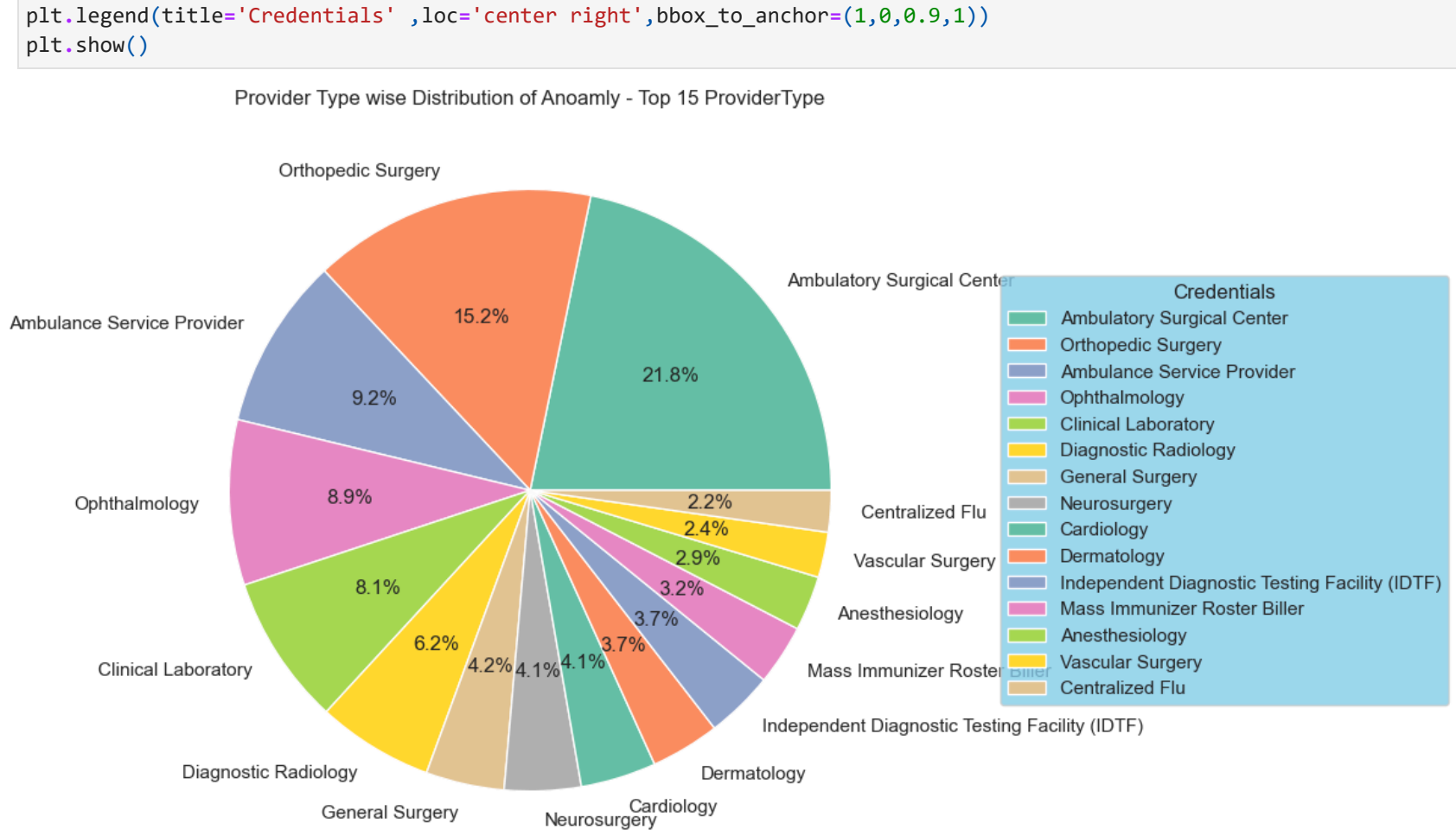
In [29]:
```
#filtering Credentials of the Provider with rows which has anomaly
Credentials_with_anomalies = df_i[df_i['Is_Anomaly']==1]['Credentials of the Provider']

# 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('Set2'))
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 **92.3%** followed by **DO 4.7%**, **CRNA 1.5%** .

In [30]:
```
#filtering ProviderType with rows which has anomaly
ProviderType_with_anomalies = df_i[df_i['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_palette('Set2'))
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()
```

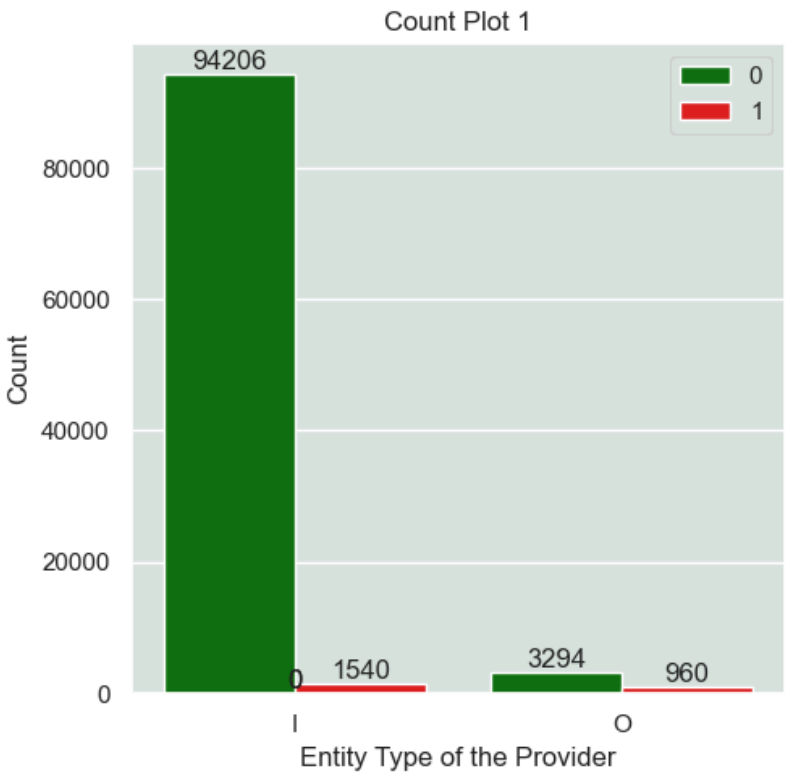


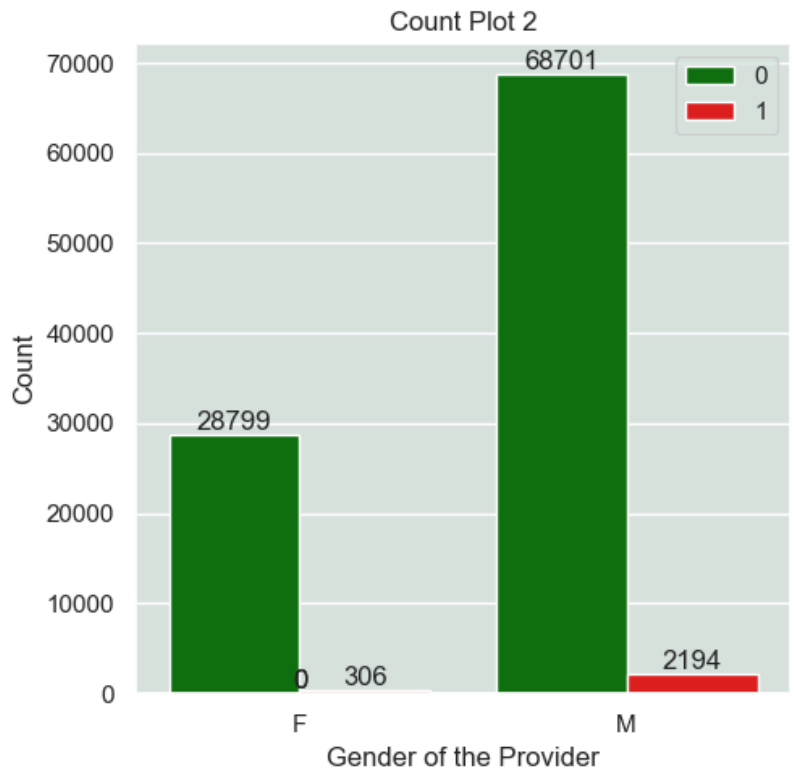
Inferences from the Pie-Chart :

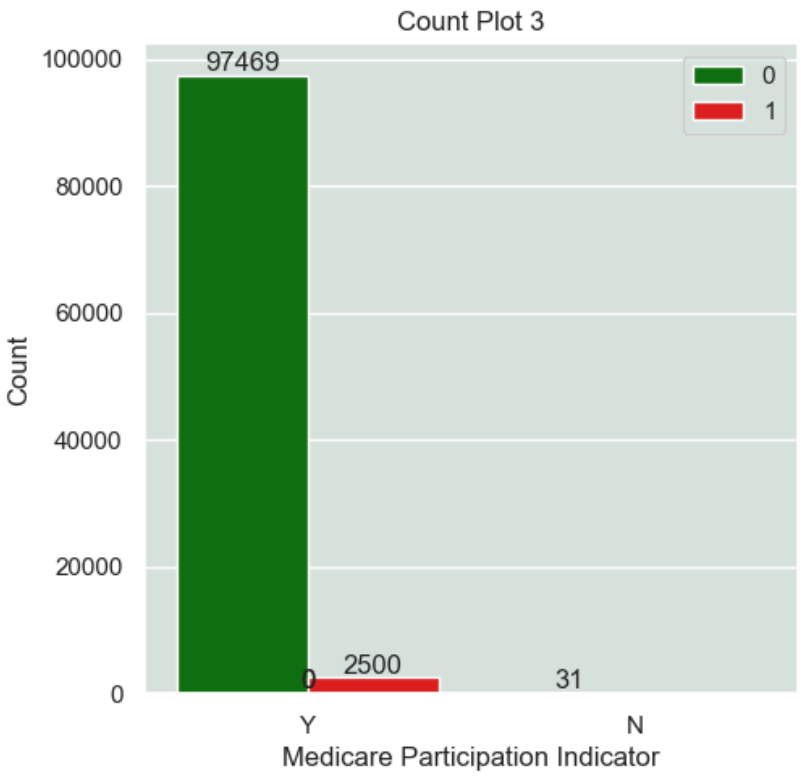
- The pie chart shows **Top 15 ProviderType** with anomaly percentage where **Ambulatory Surgical Center** has highest no of anomalies **21.8%** followed by **Ortopedic Surgery 15.2%**, **Ambulance Service Provider 9.2%** .

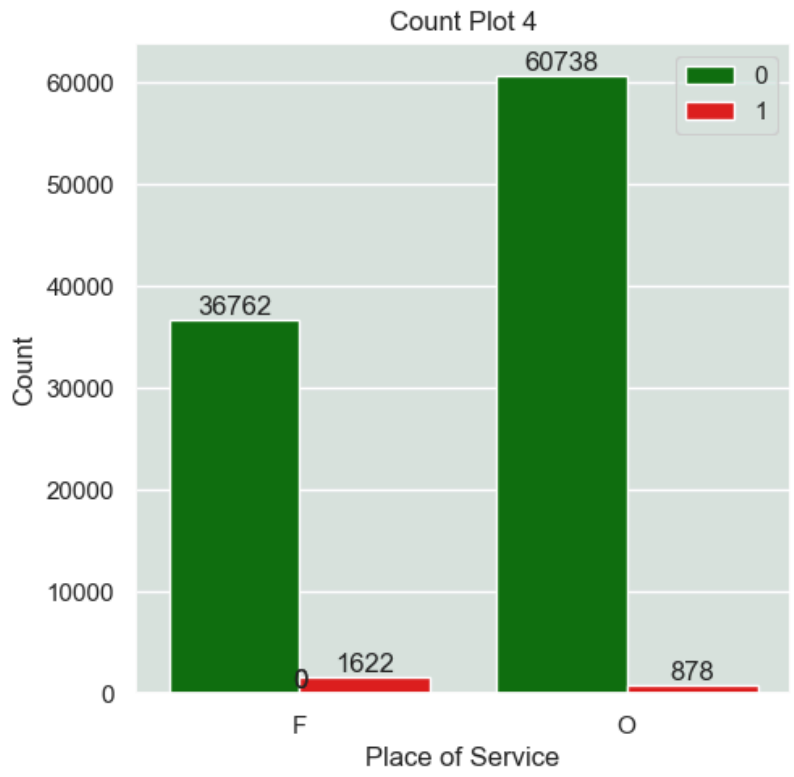
In [11]:

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

#Count plot 1
sns.countplot(x='Entity Type of the Provider',data=df_i, hue='Is_Anomaly',ax=axs[0,0],palette=['green','red'])
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_i, hue='Is_Anomaly',ax=axs[0,1],palette=['green','red'])
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_i, hue='Is_Anomaly',ax=axs[1,0],palette=['green','red'])
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_i, 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', va='bottom')

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 **1540 anomalies** where **O-organization** has **960 anomalies** only which indicates that individual entity has more fraudulent transactions.
- **Count Plot 2** shows anomaly in the **Gender** where **F-Female** has **306 anomalies** where **M-Male** has **2194 anomalies** which indicates that in Male has more fraudulent transactions.
- **Count Plot 3** shows anomaly in the **Medicare Participation Indicator** where **Y-Yes** has **2500 anomalies** where **N-No** has **0 anomalies** only which indicates that **Yes has all** fraudulent transactions.
- **Count Plot 4** shows anomaly in the **Place of Service** where **F-Facility** has **1622 anomalies** where **O-Non-Facility** has **878 anomalies** which indicates that facility has more no of fraudulent transactions.

Using Elliptic Envelope algorithm

- Given the multi-dimensional nature of the data, it would be prudent to use algorithms that can detect anomalies in multi-dimensional spaces. I am going to use the **Elliptic Envelope** algorithm for this task. Elliptic Envelope is an unsupervised machine learning algorithm used for anomaly

detection. It creates an imaginary area around the given dataset ,with values inside envelope considered as normal and anything outside labeled as outliers.

In [12]:
```
anomaly_detector = EllipticEnvelope(contamination=0.02)
anomaly_detector.fit(hdata_scaled)
df['Anomaly_Scores'] = anomaly_detector.predict(hdata_scaled)
df['Is_Anomaly'] = [1 if x == -1 else 0 for x in df['Anomaly_Scores']]
```

```
C:\Users\tmbha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\covariance\_robust_covariance.
py:747: UserWarning: The covariance matrix associated to your dataset is not full rank
  warnings.warn(
```

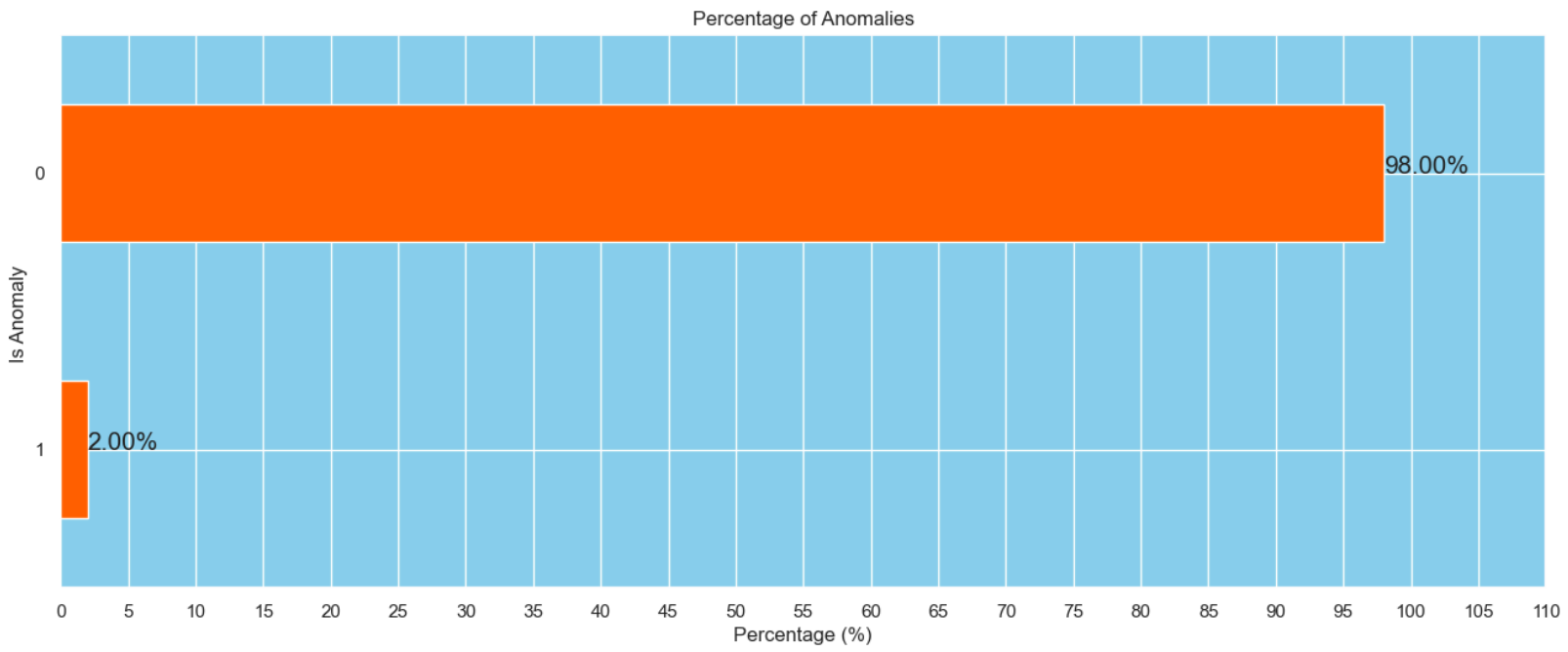
In [40]:
```
df.sample(5).T
```

Out[40]:

	73252	12791	87501	71463	96261
Credentials of the Provider	MD, MBA	MD	MD	MD	MD
Gender of the Provider	M	M	M	F	M
Entity Type of the Provider	I	I	I	I	I
City of the Provider	LUBBOCK	ROCKLEDGE	WICHITA	NEWARK	DEDHAM
State Code of the Provider	TX	FL	KS	NJ	MA
Country Code of the Provider	US	US	US	US	US
ProviderType	Rheumatology	Family Practice	Cardiology	Psychiatry	Internal Medicine
Medicare Participation Indicator	Y	Y	Y	Y	Y
Place of Service	O	O	F	F	O
HCPCS Description	Injection beneath the skin or into muscle for ...	New patient office or other outpatient visit, ...	Hospital discharge day management, 30 minutes ...	Hospital discharge day management, 30 minutes ...	Phosphate level
HCPCS Drug Indicator	N	N	N	N	N
Number of Services	36.0	73.0	83.0	20.0	14.0
Number of Medicare Beneficiaries	26	73	80	20	14
Number of Distinct Medicare Beneficiary/Per Day Services	36	73	83	20	14
Average Medicare Allowed Amount	24.4	162.39	70.23	75.5	2.765
Average Submitted Charge Amount	50.0	250.0	135.0	177.0	12.0
Average Medicare Payment Amount	17.746944	92.96589	54.396627	59.19	2.709286
Average Medicare Standardized Amount	19.353333	97.984932	57.843855	57.68	2.709286
Name	CALMES JAMES M	KOBOBEL JASEN S	REUSSER LAYNE M	CLOUDEN TOBECHUKWU	LEE DANIEL C
Full Address	5220 80TH ST	1950 ROCKLEDGE BLVD STE 101	9350 E 35TH ST N STE 101	30 BERGEN ST	1 LYONS ST
Anomaly_Scores	1	1	1	1	1
Is_Anomaly	0	0	0	0	0

In [42]:
```
# 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='#ff6200')

# 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()
```

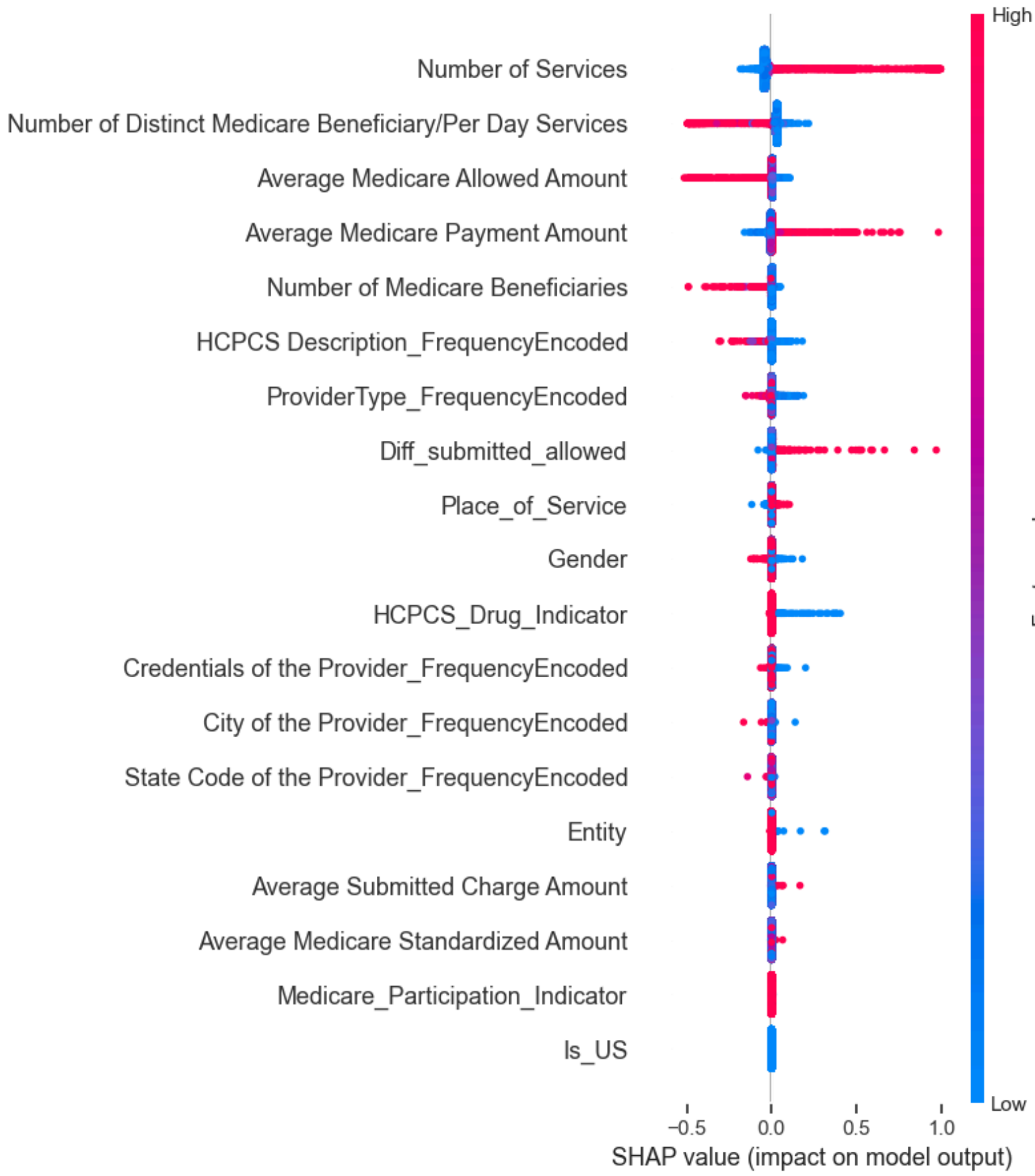


Inferences from the Graph :

- There are total **2.00%** anomalies are detected by our model i.e.around 2000 entries in entire dataset .

In [43]:
```
sns.set(rc={'axes.facecolor': 'white'}, style='darkgrid')

#creating new dataframe with scaled data and anomaly column as target variable for shap analysis
EE = hdata_scaled.join(df['Is_Anomaly'], how='inner')

#creating sample size due to system constraints
#EE_sample = EE.sample(n=10000)

#splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(EE.drop('Is_Anomaly',axis=1),EE['Is_Anomaly'],test_size=0

#training logistic regression model on data
model = LogisticRegression()
model.fit(X_train, y_train)

#creating a SHAP explainer object
explainer = shap.Explainer(model.predict,X_train)

#computing shap values for the testing data
shap_values = explainer.shap_values(X_test)

#Plotting the SHAP values using a summary plot
shap.summary_plot(shap_values, X_test, show=False)
plt.show()
```

```
PermutationExplainer explainer: 20001it [04:26, 72.73it/s]
```

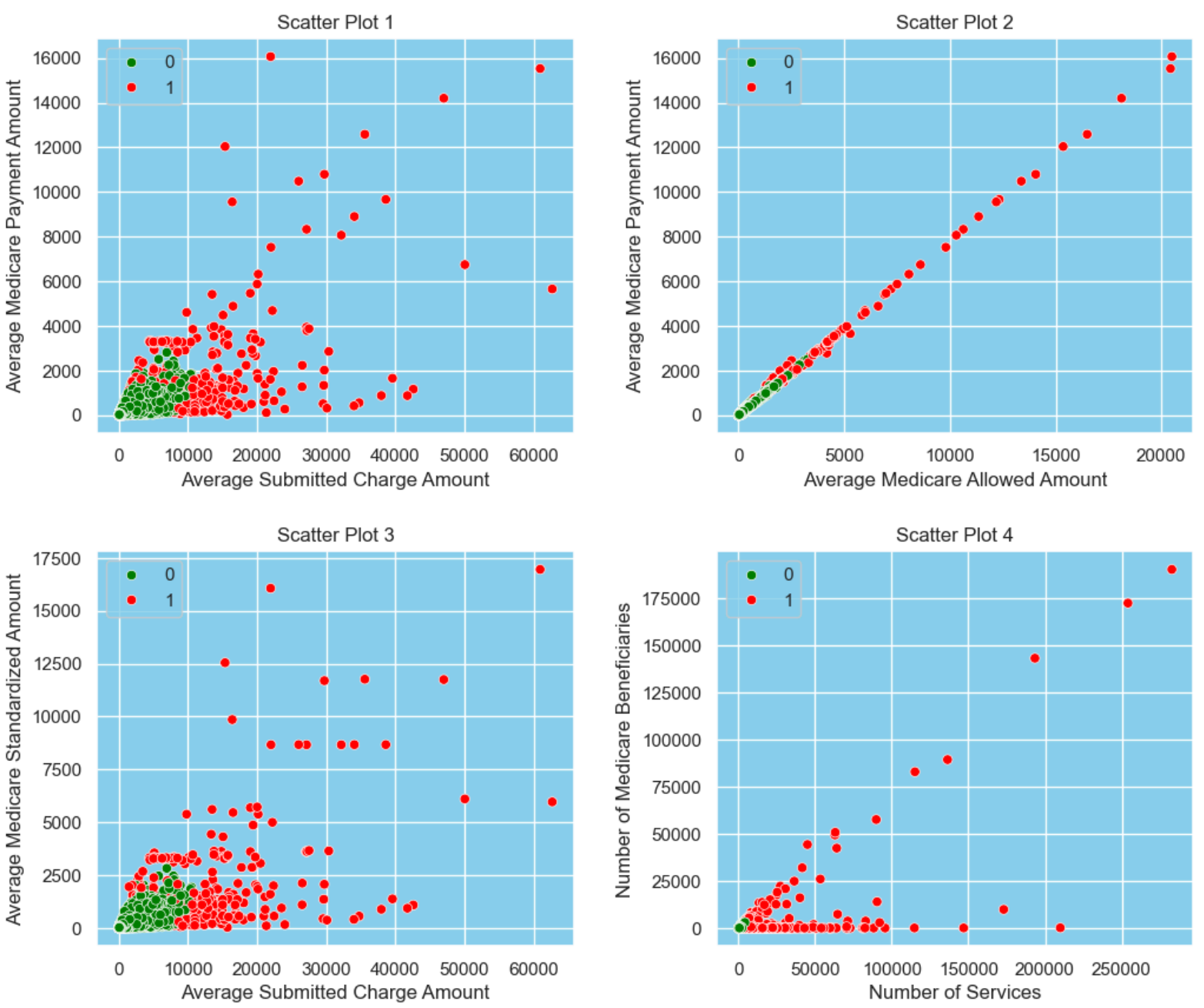


Inferences from the Summary Plot of Elliptic Envelope :

- There is major **positive** infulence of columns on model such columns are **Number of services**, **Average Medicare Payment Amount** some numerical columns are impacting model **negatively** like **number of distinct beneficiary** and **Average Medicare Allowed Amount** none of categorical columns are adding impact on model not positive or negative (neutral) or no effect .

In [36]:
```
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_Ar
                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_Ar
                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=ax
                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()
```

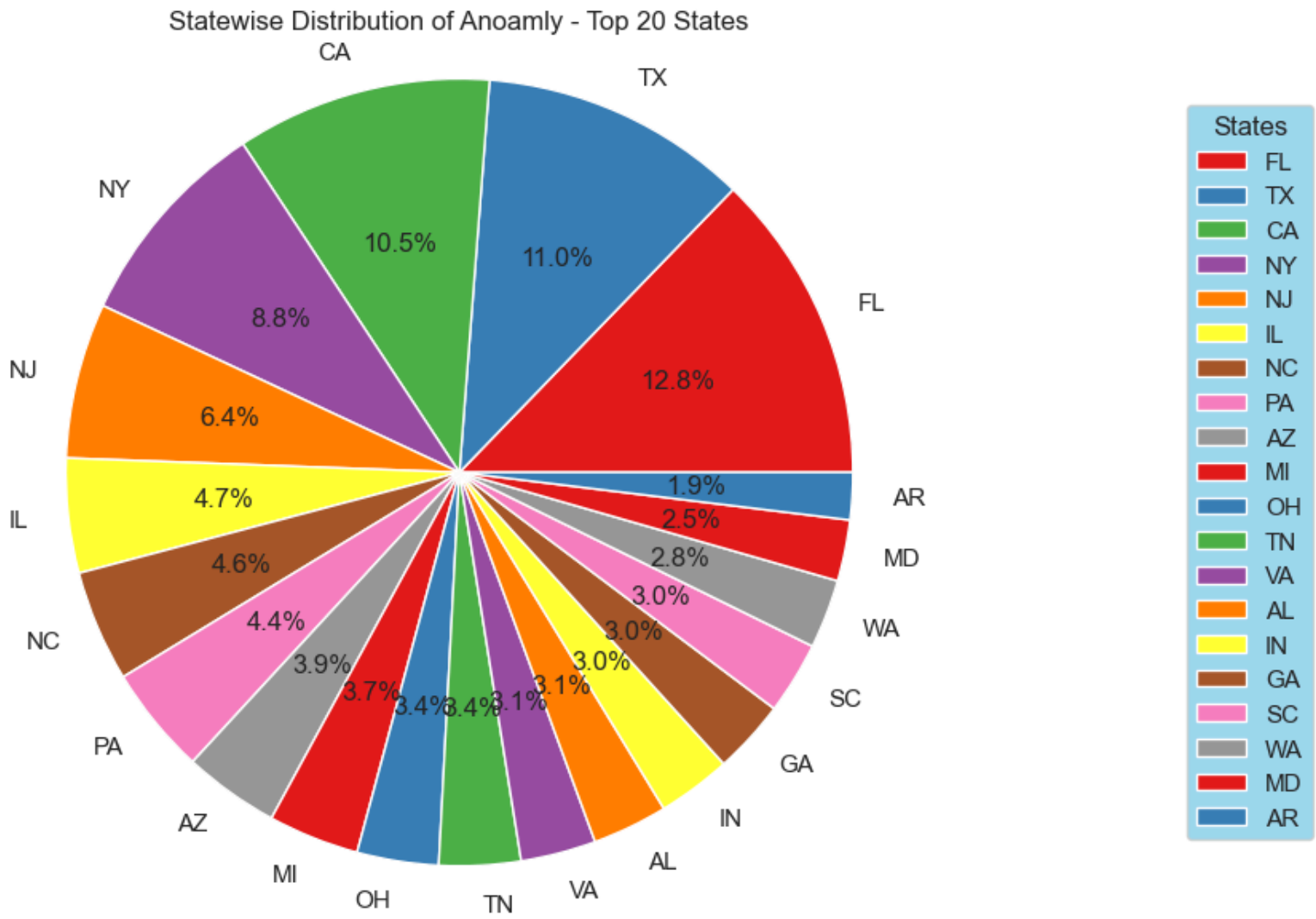


Inferences from the Scatter-Plots :

- By Visualizing the **Scatter Plots** we can see that our **Elliptic Envelope** algorithm 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 [37]:

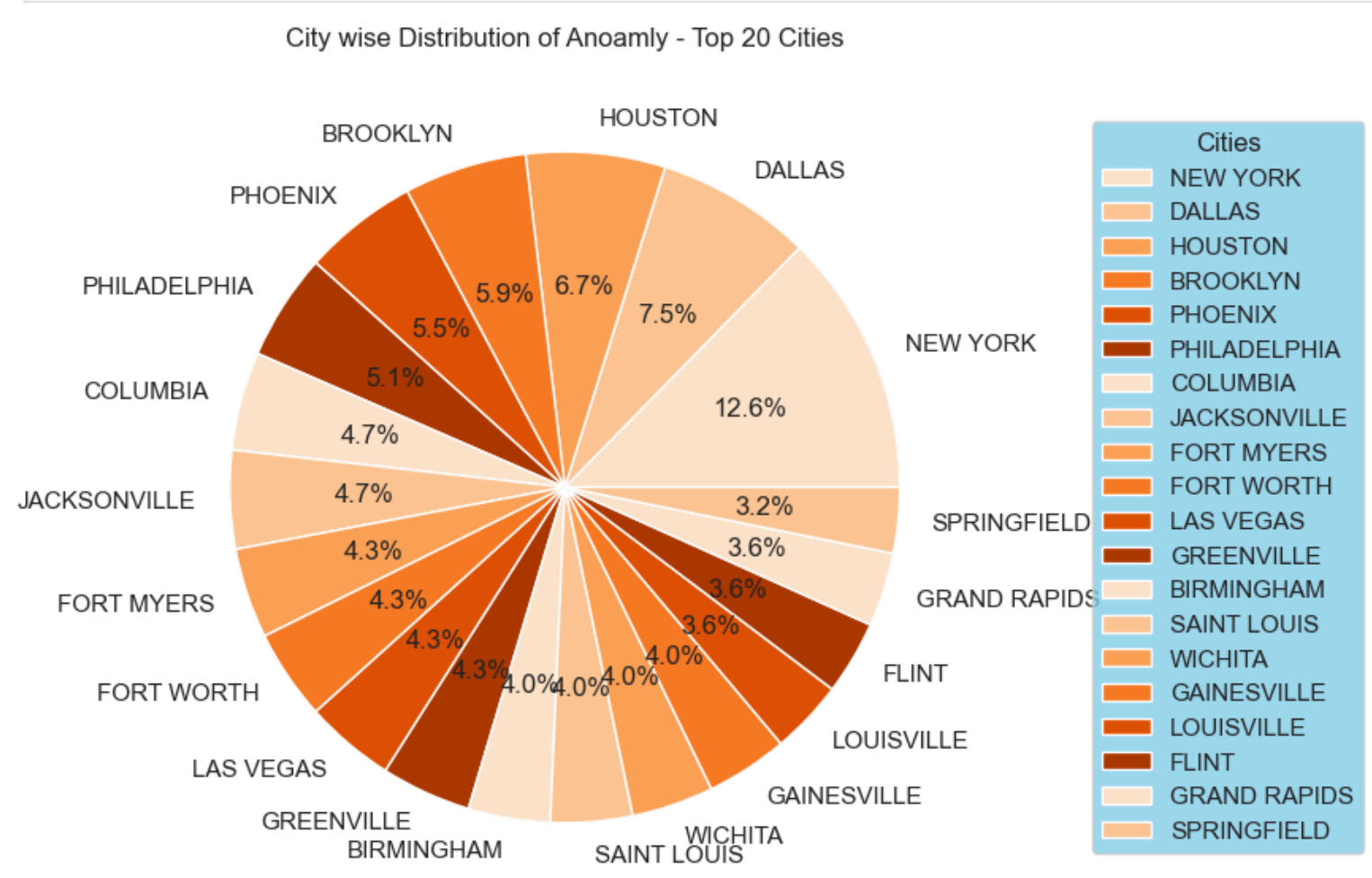
```
#filtering States with rows which has anomaly
States_with_anomalies = df[df['Is_Anomaly']==1]['State Code of the Provider']

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

#creating pie chart
plt.figure(figsize=(7,7))
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 **12.8%** followed by **Texas 11.0%**, **California 10.5%** .

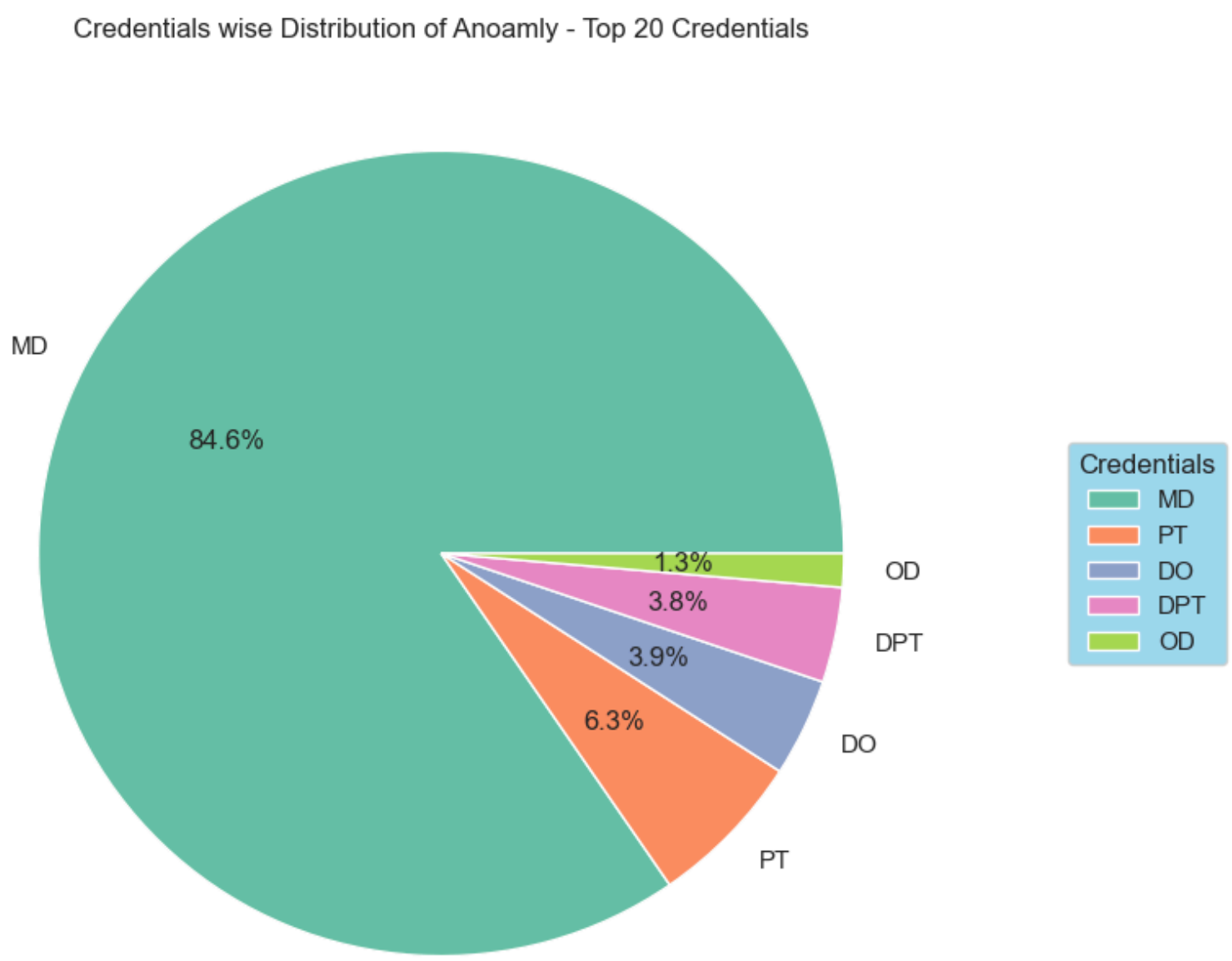
In [38]:
```
#filtering cities with rows which has anomaly
cities_with_anomalies = df[df['Is_Anomaly']==1]['City of the Provider']

# 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 **NEWYORK** has highest no of anomalies **12.6%** followed by **Dallas 7.5%**, **Houston 6.7%** .

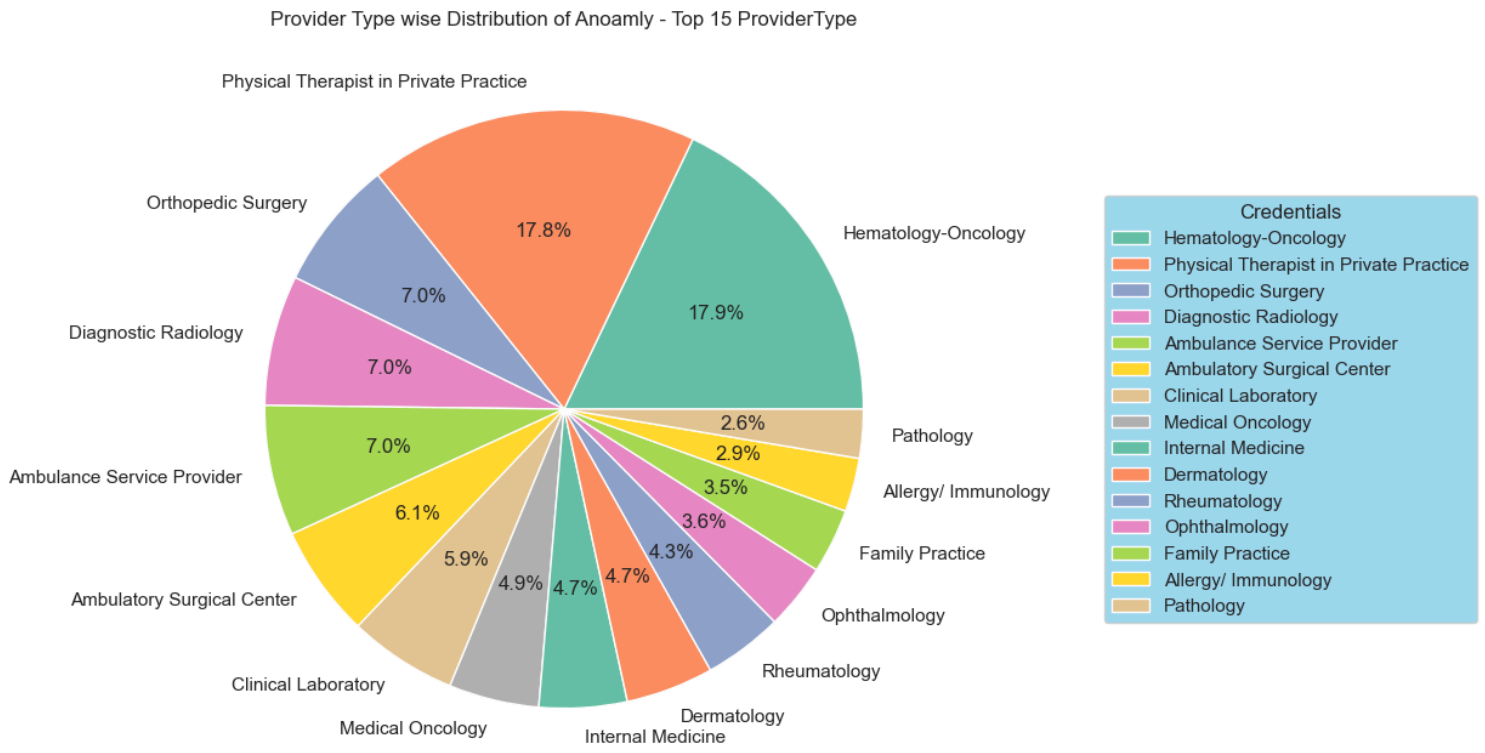
In [39]:
```
#filtering Credentials of the Provider with rows which has anomaly
Credentials_with_anomalies = df[df['Is_Anomaly']==1]['Credentials of the Provider']

# 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 **84.6%** followed by **PT 6.3%**, **DO 3.9%** .

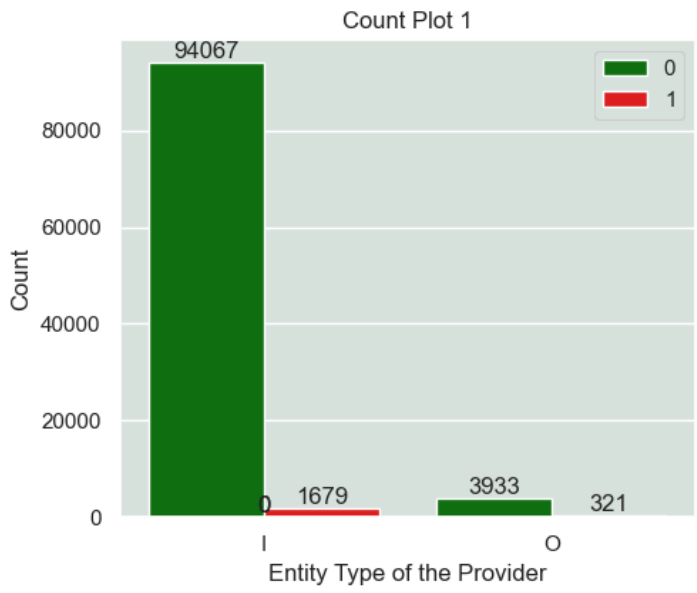
In [40]:
```
#filtering ProviderType with rows which has anomaly
ProviderType_with_anomalies = df[df['Is_Anomaly']==1]['ProviderType']

# 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_palet
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()
```

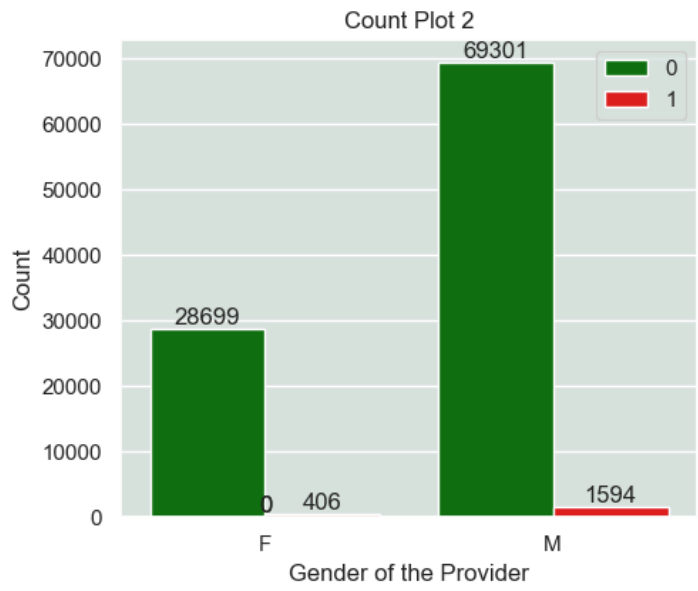


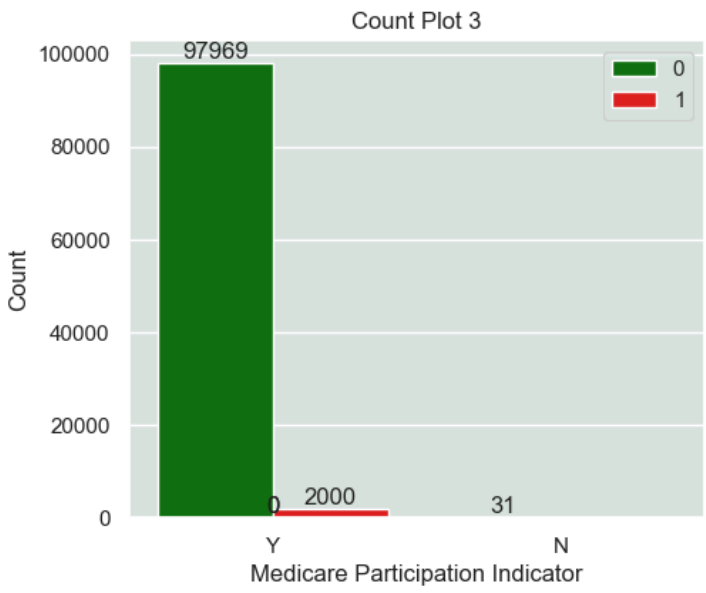
Inferences from the Pie-Chart :

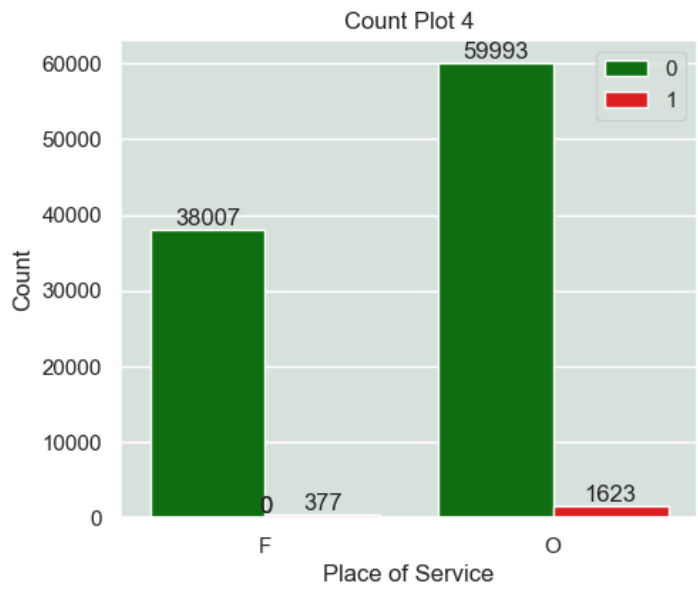
- The pie chart shows **Top 15 ProviderType** with anomaly percentage where **Hematology-Oncology** has highest no of anomalies **17.9%** followed by **Physical Therapist in Private sector 17.8%**, **Orthopedic surgery 7.0%** .

In [13]:
```
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','red'
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=['gr
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 **1679 anomalies** where **O-organization** has **321 anomalies** only which indicates that individual entity has more fraudulent transactions.
- **Count Plot 2** shows anomaly in the **Gender** where **F-Female** has **406 anomalies** where **M-Male** has **1594 anomalies** which indicates that in Male has more fraudulent transactions.
- **Count Plot 3** shows anomaly in the **Medicare Participation Indicator** where **Y-Yes** has **2000 anomalies** where **N-No** has **0 anomalies** only which indicates that Yes has all fraudulent transactions.
- **Count Plot 4** shows anomaly in the **Place of Service** where **F-Facility** has **377 anomalies** where **O-Non-Facility** has **1623 anomalies** which indicates that Non Facility Places has more no of fraudulent transactions.

Using One Class SVM algorithm

- Given the multi-dimensional nature of the data, it would be prudent to use algorithms that can detect anomalies in multi-dimensional spaces. I am going to use the **One-class SVM** algorithm for this task. This is a type of unsupervised machine learning algorithm that can be used for anomaly detection.
- In a one-class SVM, the algorithm is trained on a dataset that contains only normal or in-class samples.The goal is to find a decision boundary that separates the normal data from the rest of feature space, which is considered as anomalous.

In [14]:
```
#Defining model parameters and initialization of model
model = OneClassSVM(nu=0.035, kernel='rbf', gamma=0.1)

#fitting data to model
model.fit(hdata_scaled)
```

```
#Predicting anomalies
anomaly_scores = model.predict(hdata_scaled)

#creating new column to identify anomalies
df_s['Is_Anomaly'] = [1 if x == -1 else 0 for x in anomaly_scores]

#displaying few rows of data after prediction
df_s.sample(5).T
```

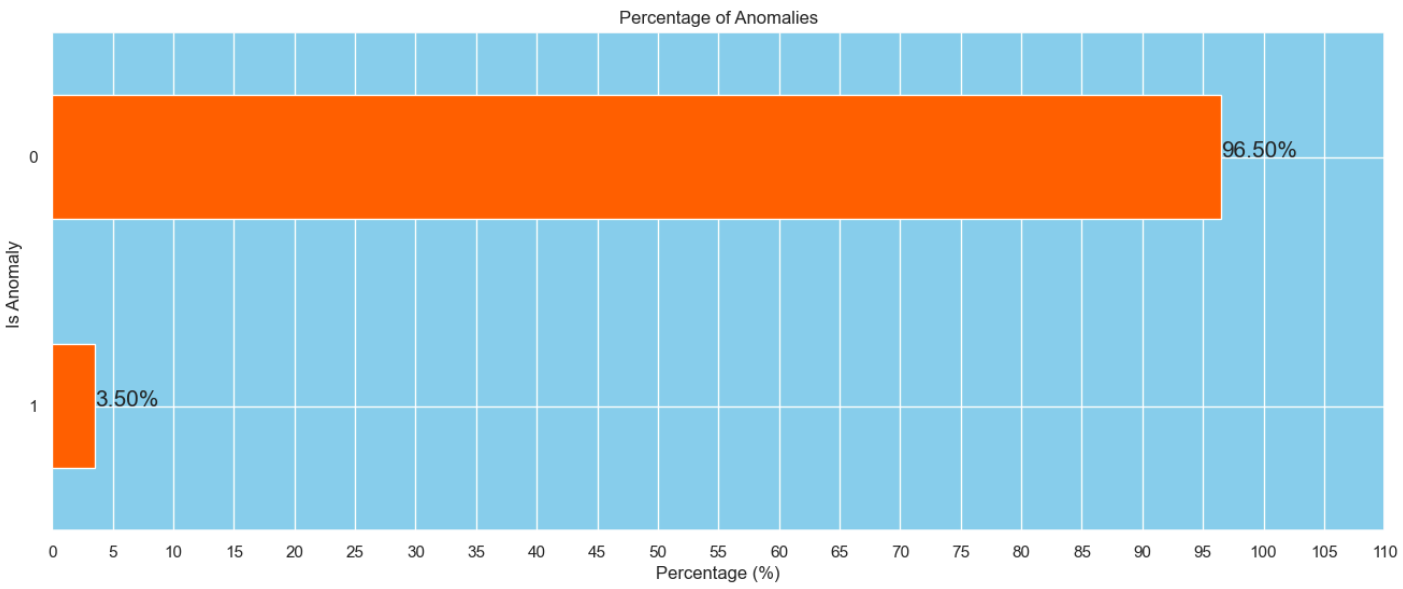
Out[14]:

	62141	19774	24432	32550	49508
Credentials of the Provider	MD	MD	PA	MD	DO
Gender of the Provider	M	M	F	M	M
Entity Type of the Provider	O	I	I	I	I
City of the Provider	BAILEYVILLE	JACKSONVILLE	LEXINGTON	LUBBOCK	JUPITER
State Code of the Provider	ME	FL	KY	TX	FL
Country Code of the Provider	US	US	US	US	US
ProviderType	Ambulance Service Provider	Diagnostic Radiology	Physician Assistant	General Surgery	Family Practice
Medicare Participation Indicator	Y	Y	Y	Y	Y
Place of Service	F	F	O	O	F
HCPCS Description	Specialty care transport (sct)	X-ray of spine, 1 view	Insertion of needle into vein for collection o...	New patient office or other outpatient visit, ...	Critical care delivery critically ill or injur...
HCPCS Drug Indicator	N	N	N	N	N
Number of Services	70.0	31.0	299.0	15.0	42.0
Number of Medicare Beneficiaries	68	31	249	15	36
Number of Distinct Medicare Beneficiary/Per Day Services	70	31	299	15	42
Average Medicare Allowed Amount	850.0	7.92	3.0	201.2	226.7
Average Submitted Charge Amount	850.0	29.0	9.986622	300.0	440.0
Average Medicare Payment Amount	666.4	5.847742	2.931639	151.144	174.18
Average Medicare Standardized Amount	593.76	6.29	2.94	166.764667	177.83
Name	WASHINGTON COUNTY EMERGENCY MEDICAL SERVICE AU...	MOON BRIAN B	ARNETT DENISE D	CUMMINS DOUGLAS G	GOLDBERG RICHARD A
Full Address	205 MAIN ST	3599 UNIVERSITY BLVD. S. BLDG. 300	2101 NICHOLASVILLE RD	4515 MARSHA SHARP FWY	2163 SOUTH U.S. HWY 1
Anomaly_Scores	1	1	1	1	1
Is_Anomaly	1	0	0	0	0

In [9]:

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

# Calculate the percentage of anomalies
anomalies_percentage = df_s['Is_Anomaly'].value_counts(normalize=True) * 100

# Plotting the percentage of anomalies
plt.figure(figsize=(16,6))
anomalies_percentage.plot(kind='barh', color='#ff6200')

# 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()
```

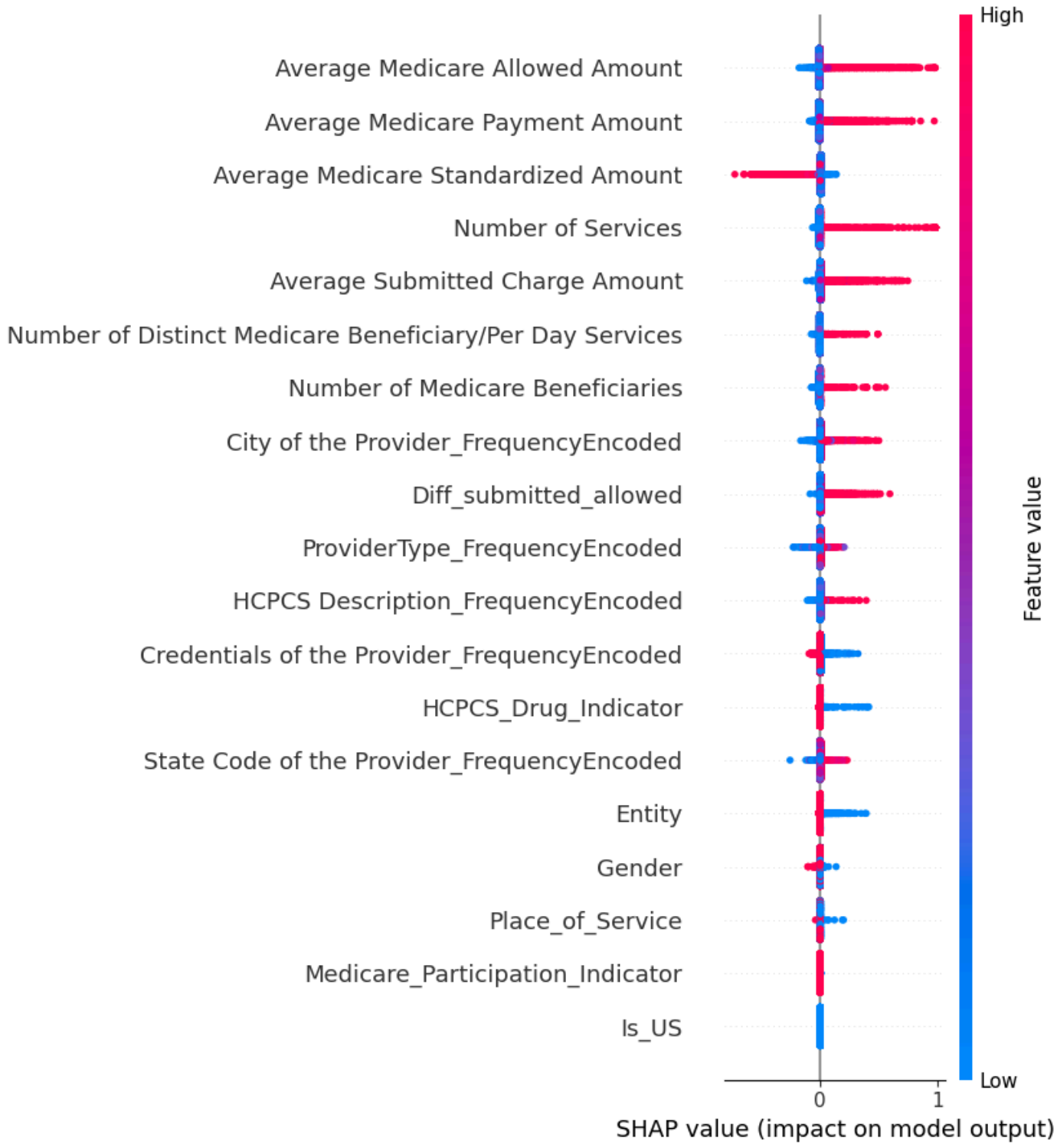


Inferences from the Graph :

- There are total **3.50%** anomalies are detected by our model i.e.around 3500 entries in entire dataset .

In [5]:
```
# sns.set(rc={'axes.facecolor': 'white'}, style='darkgrid')

#creating new dataframe with scaled data and anomaly column as target variable for shap analysi
Osvm = hdata_scaled.join(df_s['Is_Anomaly'], how='inner')

#creating sample size due to system constraints
#Osvm_sample = Osvm.sample(n=10000)

#splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(Osvm.drop('Is_Anomaly',axis=1),Osvm['Is_Ano

#training logistic regression model on data
model = LogisticRegression()
model.fit(X_train, y_train)

#creating a SHAP explainer object
explainer = shap.Explainer(model.predict,X_train)

#computing shap values for the testing data
shap_values = explainer.shap_values(X_test)

#Plotting the SHAP values using a summary plot
shap.summary_plot(shap_values, X_test, show=False)
plt.show()
```

```
PermutationExplainer explainer: 20001it [11:54, 27.60it/s]
```

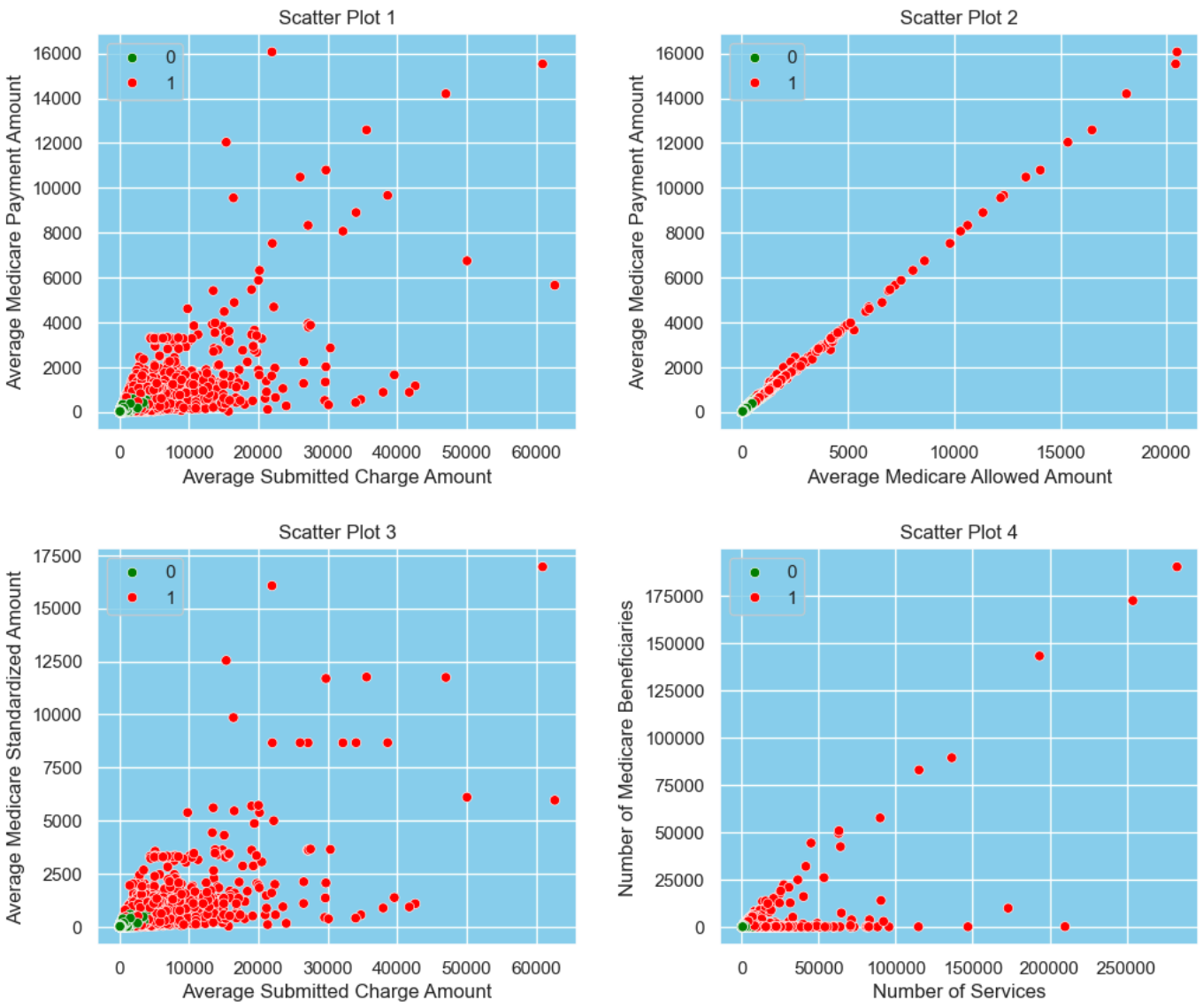


Inferences from the Summary Plot of One Class SVM :

- There is major **positive** infulence of **Numerical** columns on model except column **Average Medicare Standardized Amount** which impacting model **negatively** some of categorical columns are adding impact on model **positively** like **HCPCS description**, **city** and **Provider Type** .

In [10]:
```
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=
                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=
                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',
                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_s, hue='I
                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()
```

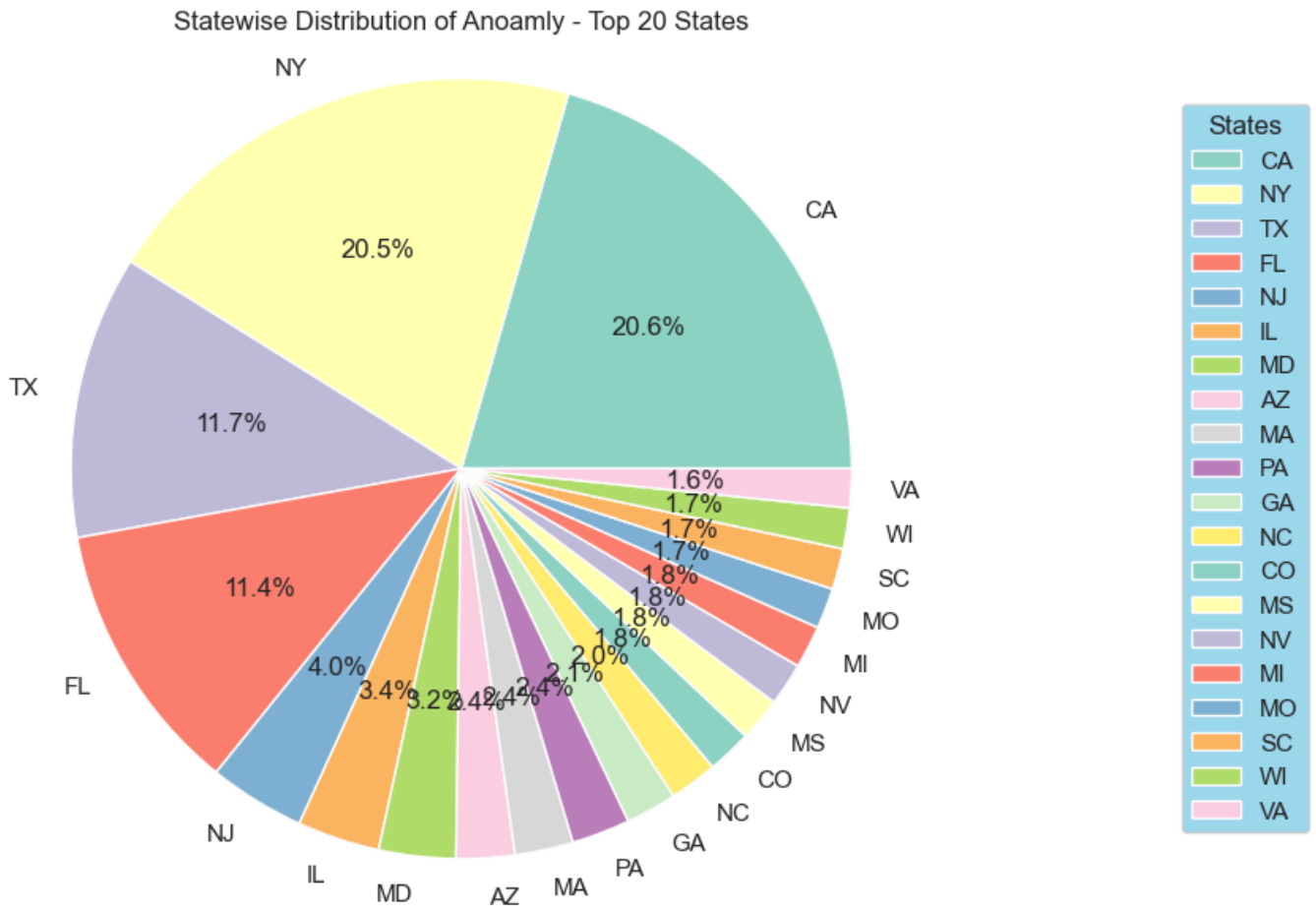
```
plt.subplots_adjust(wspace=0.3,hspace=0.3)

plt.show()
```



Inferences from the Scatter-Plots :

- By Visualizing the **Scatter Plots** we can see that our **One-class SVM** algorithm 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**.

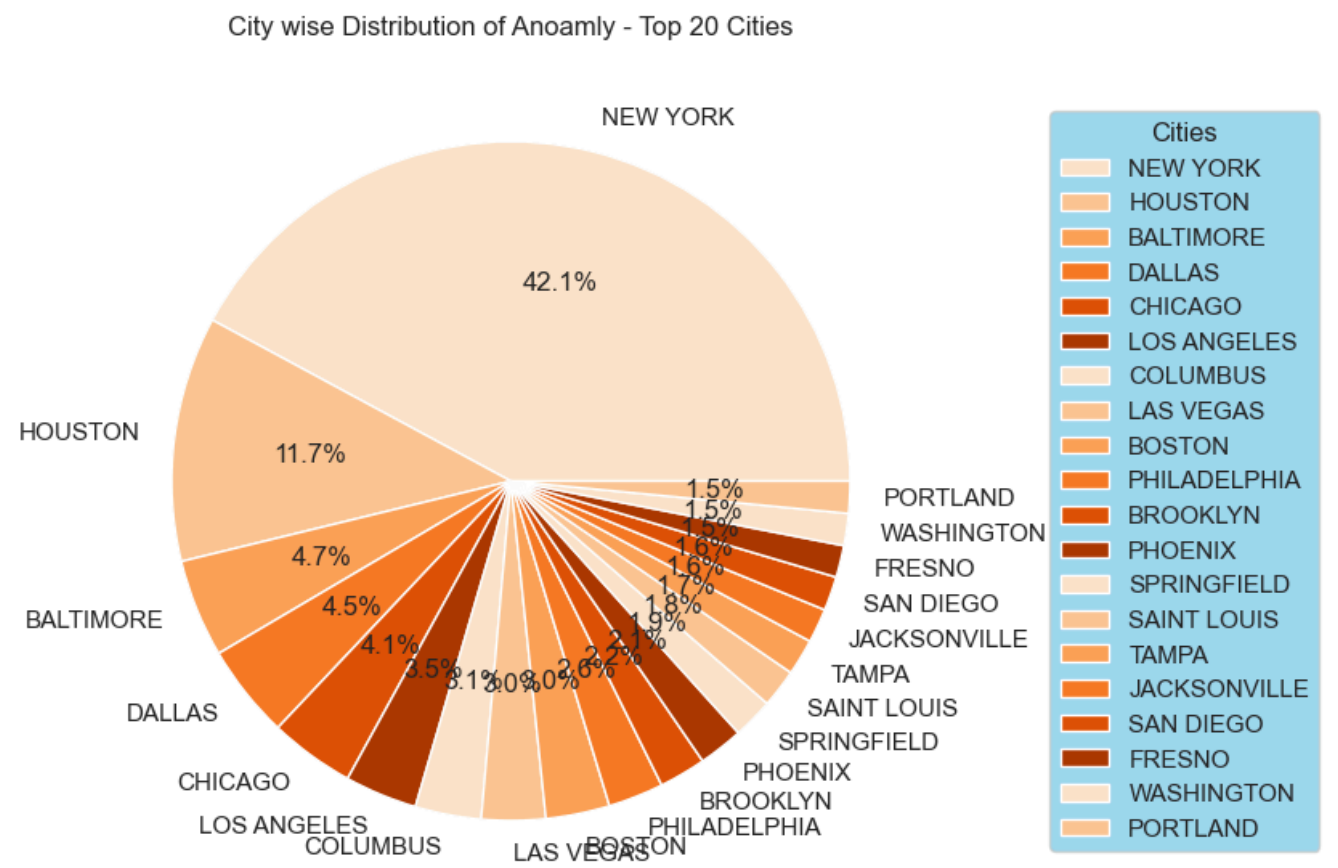
In [11]:
```
#filtering States with rows which has anomaly
States_with_anomalies = df_s[df_s['Is_Anomaly']==1]['State Code of the Provider']

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

#creating pie chart
plt.figure(figsize=(7,7))
plt.pie(State_counts,labels=State_counts.index, autopct='%1.1f%%',colors=sns.color_palette('
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 **California** has highest no of anomalies **20.6%** followed by **New York 20.5%**,and **Texas 11.7%** .

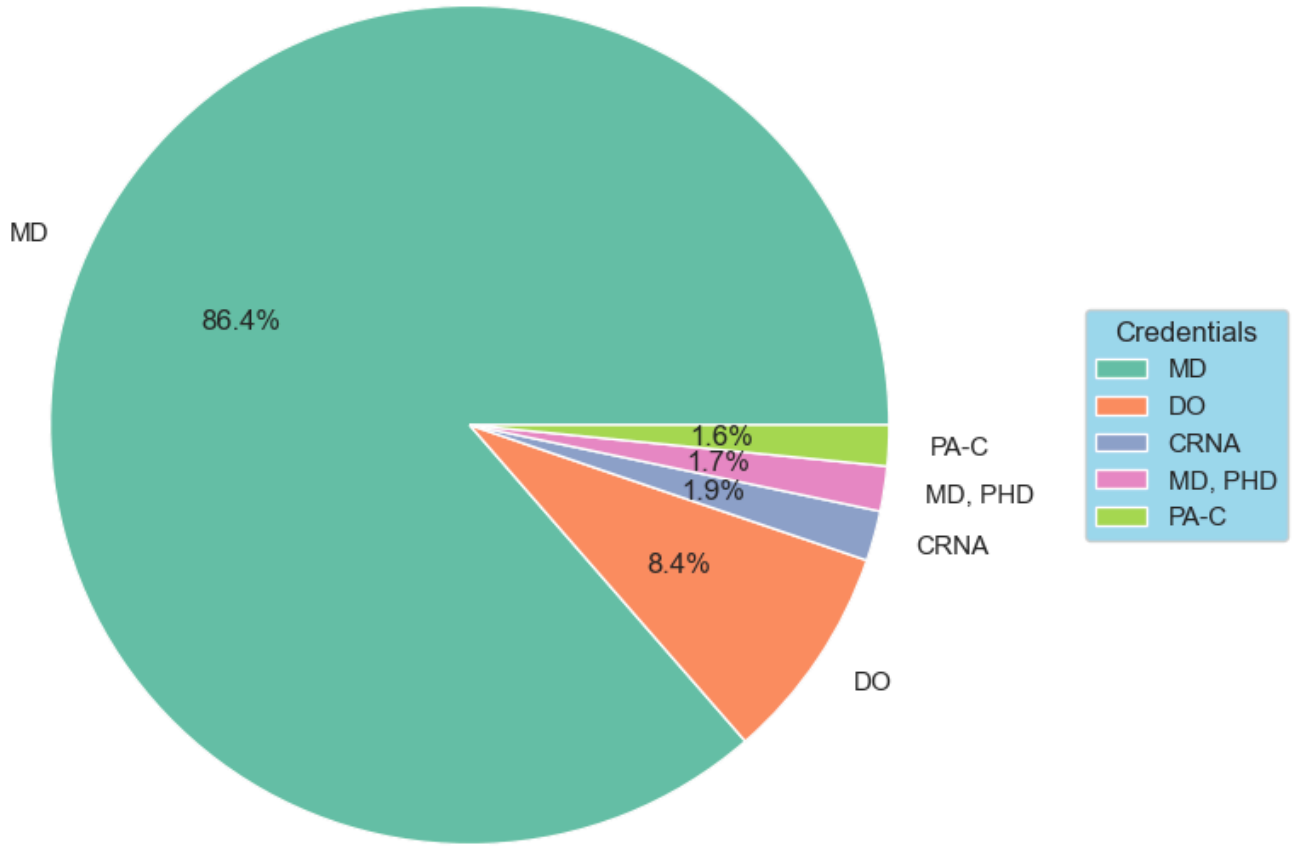
In [12]:
```
#filtering cities with rows which has anomaly
cities_with_anomalies = df_s[df_s['Is_Anomaly']==1]['City of the Provider']

# 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('C
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 **NEWYORK** has highest no of anomalies **42.1%** followed by **Houston 11.7%**, **Baltimore 4.7%** .

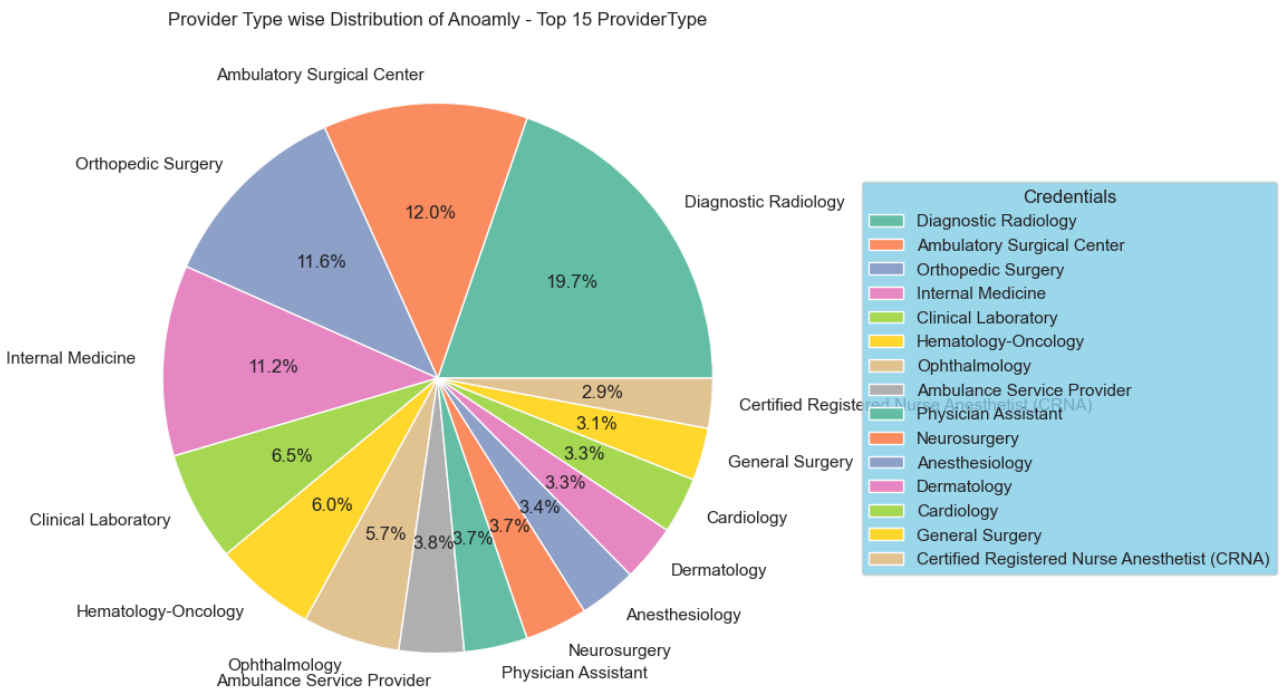
In [13]:
```
#filtering Credentials of the Provider with rows which has anomaly
Credentials_with_anomalies = df_s[df_s['Is_Anomaly']==1]['Credentials of the Provider']

# 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.co
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 **86.4%** followed by **DO 8.4%**, **CRNA 1.9%** .

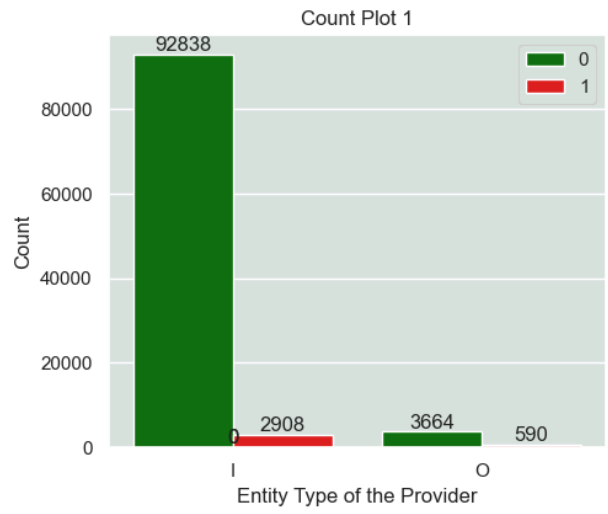
In [14]:
```
#filtering ProviderType with rows which has anomaly
ProviderType_with_anomalies = df_s[df_s['Is_Anomaly']==1]['ProviderType']

# 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=s
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()
```

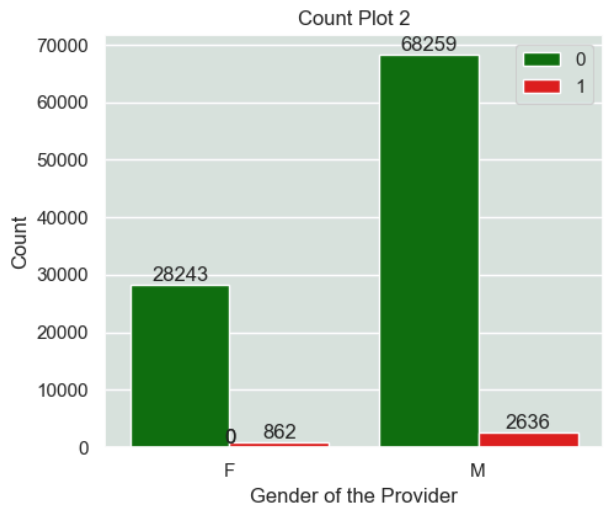


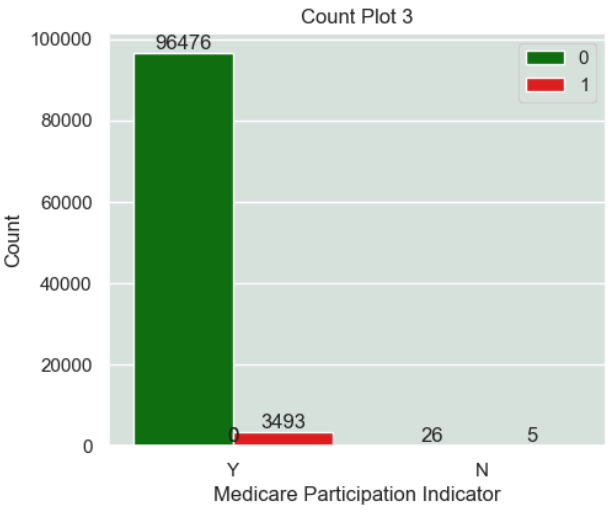
Inferences from the Pie-Chart :

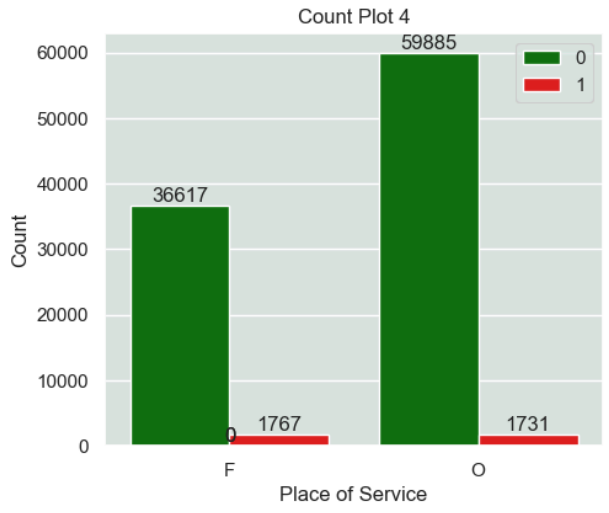
- The pie chart shows **Top 15 ProviderType** with anomaly percentage where **Diagnostic Radiology** has highest no of anomalies **19.7%** followed by **Ambulatory Surgical center 12.0%**, **Orthopedic surgery 11.6%** .

In [15]:
```
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_s, hue='Is_Anomaly',ax=axs[0,0],p
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_s, hue='Is_Anomaly',ax=axs[0,1],palett
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_s, hue='Is_Anomaly',ax=axs[1
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_s, hue='Is_Anomaly',ax=axs[1,1],palette=['gr
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()

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

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())

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 **2908 anomalies** where **O-organization** has **590 anomalies** only which indicates that individual entity has more fraudulent transactions.
- **Count Plot 2** shows anomaly in the **Gender** where **F-Female** has **862 anomalies** where **M-Male** has **2636 anomalies** which indicates that in Male has more fraudulent transactions.
- **Count Plot 3** shows anomaly in the **Medicare Participation Indicator** where **Y-Yes** has **3493 anomalies** where **N-No** has **5 anomalies** only which indicates that Yes has more fraudulent transactions.
- **Count Plot 4** shows anomaly in the **Place of Service** where **F-Facility** has **1767 anomalies** where **O-Non-Facility** has **1731 anomalies** which indicates that both Places has equal no of fraudulent transactions.