Preprocessing on the data set:

Identifying the missing values :

```
import pandas as pd

# Load the dataset
file_path = '/content/Healthcare Providers.csv'
data = pd.read_csv(file_path)

# Display the first few rows and summary information of the dataset
data.head(), data.info()
```

Show hidden output

Missing values with their percentages:

```
missing_values = data.isnull().mean() * 100

# Display columns with missing values
missing_values[missing_values > 0]
```

```
First Name of the Provider           4.255
Middle Initial of the Provider      29.336
Street Address 2 of the Provider    59.363
dtype: float64
```

```
df.columns
```

Show hidden output

Removing the special characters like "(",".",extra spce :

```
# removing  "(" from the column Middle Initial of the Provider

df['Middle Initial of the Provider'] = df['Middle Initial of the Provider'].str.replace('(', '')
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 64 columns

```
# removing  the extra space after the dot character in First Name of the Provider and Street Address 1 of the Provider columns

def remove_extra_space_after_dot(text):
  if isinstance(text, str):
    return text.replace('. ', '.')
  else:
    return text

df['First Name of the Provider'] = df['First Name of the Provider'].apply(remove_extra_space_after_dot)
df['Street Address 1 of the Provider'] = df['Street Address 1 of the Provider'].apply(remove_extra_space_after_dot)

df
```

Show hidden output

```
# missing value columns and their percentage

missing_values = df.isnull().mean() * 100
missing_values_dict = dict(missing_values)

# Print the columns with missing values and their percentages
for column, percentage in missing_values_dict.items():
  if percentage > 0:
    print(f"Column: {column}, Missing Values: {percentage:.2f}%")
```

```
Column: Middle Initial of the Provider, Missing Values: 29.34%
Column: Street Address 2 of the Provider, Missing Values: 59.36%
```

```
# replace the missing values in First Name of the Provider  with "NA"

df['First Name of the Provider'] = df['First Name of the Provider'].fillna('NA')
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 64 columns

```
df.isnull().sum()
```

```
index                                                     0
National Provider Identifier                              0
Last Name/Organization Name of the Provider               0
First Name of the Provider                                0
Middle Initial of the Provider                        29336
                                                      ...
Number of Distinct Medicare Beneficiary/Per Day Services  0
Average Medicare Allowed Amount                           0
Average Submitted Charge Amount                           0
Average Medicare Payment Amount                           0
Average Medicare Standardized Amount                      0
Length: 64, dtype: int64
```

```
df.isnull().sum()
```

```
index                                                     0
National Provider Identifier                              0
Last Name/Organization Name of the Provider               0
First Name of the Provider                                0
Middle Initial of the Provider                        29336
                                                      ...
Number of Distinct Medicare Beneficiary/Per Day Services  0
Average Medicare Allowed Amount                           0
Average Submitted Charge Amount                           0
Average Medicare Payment Amount                           0
Average Medicare Standardized Amount                      0
Length: 64, dtype: int64
```

```
#  replaceing  the missing values in Middle Initial of the Provider  with mode of the column

mode_value = df['Middle Initial of the Provider'].mode()[0]
df['Middle Initial of the Provider'] = df['Middle Initial of the Provider'].fillna(mode_value)
df
```

Show hidden output

```
df.isnull().sum()
```

Show hidden output

performing encoding on Street Address 2 of the Provider and fill the missing values in this column with mean

```
# performing encoding on Street Address 2 of the Provider  and fill the missing values in this column with mean

from sklearn.preprocessing import LabelEncoder

# Encode Street Address 2 of the Provider
le = LabelEncoder()
df['Street Address 2 of the Provider'] = le.fit_transform(df['Street Address 2 of the Provider'].fillna('Unknown'))

# Fill missing values in Street Address 2 of the Provider with the mean
df['Street Address 2 of the Provider'].fillna(df['Street Address 2 of the Provider'].mean(), inplace=True)
```

Double-click (or enter) to edit

```
null_values = df.isnull().sum().sum()
print(f"Number of null values: {null_values}")
```

```
Number of null values: 0
```

```
df.isnull().sum()
```

```
index                                                     0
National Provider Identifier                              0
Last Name/Organization Name of the Provider               0
First Name of the Provider                                0
Middle Initial of the Provider                            0
```

```
                                                    ..
Number of Distinct Medicare Beneficiary/Per Day Services    0
Average Medicare Allowed Amount                             0
Average Submitted Charge Amount                             0
Average Medicare Payment Amount                             0
Average Medicare Standardized Amount                        0
Length: 64, dtype: int64
```

Converting Zip Code of the Provider to a string

```
#  Convert Zip Code of the Provider to a string

df['Zip Code of the Provider'] = df['Zip Code of the Provider'].astype(str)
```

```
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider	Gender of the Provider	Entity Type of the Provider	Street Address 1 of the Provider	Street Address 2 of the Provider	...	HCP(Cod
0	8774979	1891106191	UPADHYAYULA	SATYASREE	A	MD	F	I	1402 S GRAND BLVD	3452	...	992
1	3354385	1346202256	JONES	WENDY	P	MD	F	I	2950 VILLAGE DR	9789	...	G02(
2	3001884	1306820956	DUROCHER	RICHARD	W	DPM	M	I	20 WASHINGTON AVE	6156	...	993
3	7594822	1770523540	FULLARD	JASPER	A	MD	M	I	5746 N BROADWAY ST	9789	...	810(
4	746159	1073627758	PERROTTI	ANTHONY	E	DO	M	I	875 MILITARY TRL	7639	...	963
...	...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN	A	PT	F	I	324 E BALTIMORE ST	9789	...	971(
99996	2079360	1215091327	HAYNER	MARGARET	S	ARNP	F	I	645 NW 4TH ST	9789	...	992
99997	8927965	1902868185	VALENCIA	DANA	A	MD	M	I	3009 N BALLAS RD	7692	...	933
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	A	NA	F	I	2365 BOSTON POST RD	7668	...	G00(
99999	3547535	1356772156	RAMEZANI	ELIIAN	A	NA	F	I	444 MIDDLE NECK RD APT 1H	9789	...	971

100000 rows × 27 columns

converting column Average Medicare Allowed Amount are in float format.

```
# converting  column   Average Medicare Allowed Amount are in float format.

df['Average Medicare Allowed Amount'] = df['Average Medicare Allowed Amount'].astype(float)
```

```
df
```

Show hidden output

performing the Binary encoding :

```
import pandas as pd
import category_encoders as ce

# Load the dataset
file_path = '/content/Healthcare Providers.csv'
data = pd.read_csv(file_path)

# List of columns to encode
columns_to_encode = [
    'Gender of the Provider',
    'Entity Type of the Provider',
    'State Code of the Provider',
    'Country Code of the Provider',
    'Provider Type',
    'City of the Provider',
    'Credentials of the Provider'
]

# Initialize the binary encoder
encoder = ce.BinaryEncoder(cols=columns_to_encode, return_df=True)

# Fit and transform the data
data_encoded = encoder.fit_transform(data)

# Save the encoded dataset
data_encoded.to_csv('/content/Healthcare Providers.csv', index=False)
```

```
# print number of rows AND COLUMNS IN  /content/Healthcare Providers.csv

import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv('/content/Healthcare Providers.csv')

# Print the number of rows and columns
print(f"Number of rows: {df.shape[0]}")
print(f"Number of columns: {df.shape[1]}")
```

```
Number of rows: 100000
Number of columns: 64
```

removing the duplicate rows

```
duplicate_rows = df[df.duplicated()]

print("Number of duplicate rows:", len(duplicate_rows))

# Print the duplicate rows
print(duplicate_rows)
```

```
Number of duplicate rows: 0
Empty DataFrame
Columns: [index, National Provider Identifier, Last Name/Organization Name of the Provider, First Name of the Provider, Middle Initial
Index: []

[0 rows x 64 columns]
```

```
print(df.shape)
```

```
(100000, 64)
```

normalization and handle missing values:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Load the dataset
file_path = '/content/Healthcare Providers.csv'
data = pd.read_csv(file_path)

# Columns to normalize
columns_to_normalize = [
    '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'
]

# Convert columns to numeric, handling errors
for col in columns_to_normalize:
    data[col] = pd.to_numeric(data[col], errors='coerce')  # 'coerce' will replace non-numeric values with NaN

# Initialize the scaler
scaler = MinMaxScaler()

# Apply normalization and handle missing values
data[columns_to_normalize] = scaler.fit_transform(data[columns_to_normalize].fillna(data[columns_to_normalize].mean()))
# Fill missing values with the mean of each column before normalization

# Save the normalized dataset
data.to_csv('/content/Healthcare Providers.csv', index=False)
```

```
df
```

Show hidden output

```
df.isnull().sum()
```

Show hidden output

Double-click (or enter) to edit

> Calculating the z-score for each data point. Handle outliers by replacing them with the mean

```
#  Identify and handle outliers in columns with continuous data to ensure they do not affect the model performance adversely.

# Define columns with continuous data
continuous_columns = ['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']

# Calculate the z-score for each data point in the continuous columns
z_scores = df[continuous_columns].apply(lambda x: (x - x.mean()) / x.std())

# Define a threshold for outlier detection
threshold = 3

# Identify outlier indices
outlier_indices = (z_scores > threshold).any(axis=1)


# Handle outliers by replacing them with the mean
for col in continuous_columns:
    df[col] = df[col].where(~outlier_indices, df[col].mean())



df
```

redentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	...	HCPCS Code	HCPCS Description	HCPCS Drug Indicator	Number of Services	Number of Medicare Beneficiaries	Numbe Dist Medi Beneficiary, Day Serv
0	0	0	0	0	...	99223	Initial hospital inpatient care typically 70 m...	N	0.052379	0.035104	0.04
0	0	0	0	0	...	G0202	Screening mammography bilateral 2view study of...	N	0.059012	0.038963	0.05
0	0	0	0	0	...	99348	Established patient home visit typically 25 mi...	N	0.021255	0.002024	0.02
0	0	0	0	0	...	81002	Urinalysis manual test	N	0.009109	0.007085	0.00
0	0	0	0	0	...	96372	Injection beneath the skin or into muscle for ...	N	0.022267	0.013158	0.02
...	...	...	...	...	...	...	...	...	...	...	
0	0	0	0	0	...	97162	Evaluation of physical therapy typically 30 mi...	N	0.009109	0.009109	0.00
0	0	0	0	0	...	99213	Established patient office or other outpatient...	N	0.126518	0.097166	0.12
0	0	0	0	0	...	93320	Doppler ultrasound study of heart blood flow v...	N	0.000000	0.000000	0.00
1	1	1	0	0	...	G0008	Administration of influenza virus vaccine	N	0.001012	0.001012	0.00
1	1	1	0	0	...	97112	Therapeutic procedure to reeducate braintonerv...	N	0.090663	0.058936	0.08

```
import pandas as pd # Import the pandas library

# Load your data into a DataFrame named 'df'
# For example, if your data is in a CSV file:
df = pd.read_csv('/content/_preprocessed and cleaned healthcare providers (1).csv')

# Now you can proceed with your code to combine the names:
df['Full Name'] = df['First Name of the Provider'] + ' ' + df['Middle Initial of the Provider'] + '. ' + df['Last Name/Organization Name of
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 65 columns

adding the new column as "Service Utilization Rate" which is obtained by the ratio of Number of Medicare Beneficiaries and Number of Services'

```
#  Calculate the
#  add the rate of service utilization  column (i.e., Number of Medicare Beneficiaries / Number of Services).

df['Service Utilization Rate'] = df['Number of Medicare Beneficiaries'] / df['Number of Services']
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 66 columns

Adding the new column as "Average Cost per Service" which is obtained by the ratio of "Average Submitted Charge Amount" and "Number of Services"

```
# prompt: Average Cost per Service: Calculate
#  add the average cost per service column (i.e., Average Submitted Charge Amount / Number of Services).

df['Average Cost per Service'] = df['Average Submitted Charge Amount'] / df['Number of Services']
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 67 columns

adding the new column column Payment to Charge Ratio: which is obtained by the ratio of the average Medicare payment amount to the average submitted charge amount.

```
# prompt: add the column Payment to Charge Ratio: Calculate the ratio of the average Medicare payment amount to the average submitted charg

df['Payment to Charge Ratio'] = df['Average Medicare Payment Amount'] / df['Average Submitted Charge Amount']
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 68 columns

```
# prompt: add the column Payment to Charge Ratio: Calculate the ratio of the average Medicare payment amount to the average submitted charg

df['Payment to Charge Ratio'] = df['Average Medicare Payment Amount'] / df['Average Submitted Charge Amount']
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 68 columns

Double-click (or enter) to edit

adding the column Total Services: Aggregate the total number of services across all providers.

```
# prompt: add the column Total Services: Aggregate the total number of services across all providers.

df['Total Services'] = df['Number of Services'].sum()
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 69 columns

```
# prompt: adding  the column Provider Location: Concatenate City of the Provider_1', 'City of the Provider_2',
#        'City of the Provider_3', 'City of the Provider_4',
#        'City of the Provider_5', 'City of the Provider_6',
#        'City of the Provider_7', 'City of the Provider_8',
#        'City of the Provider_9', 'City of the Provider_10',
#        'City of the Provider_11', 'City of the Provider_12',


df['Provider Location'] = df['City of the Provider_1'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_2'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_3'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_4'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_5'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_6'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_7'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_8'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_9'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_10'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_11'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_12'].astype(str).fillna('')
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER	NaN	0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN	NaN	0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA	NaN	0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA	NaN	1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN	NaN	1	1	1	0	0	

100000 rows × 70 columns

```
df.columns
```

```
df.shape
```

```
(100000, 65)
```

```
# prompt: fill the null values in the Middle Initial of the Provider with empty strings
import pandas as pd  # Import the pandas library for working with DataFrames

# Replace 'your_data.csv' with the actual path to your file
df = pd.read_csv('/content/_preprocessed and cleaned healthcare providers (1).csv')

# Now you can fill the null values
df['Middle Initial of the Provider'].fillna('', inplace=True)
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE		0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER		0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN		0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA		0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA		1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN		1	1	1	0	0	

100000 rows × 64 columns

```
# prompt: add full name column by con catinating First Name of the Provider,Last Name/Organization Name of the Provider,Middle Initial of t

df['Full Name'] = df['First Name of the Provider'] + ' ' + df['Middle Initial of the Provider'] + '. ' + df['Last Name/Organization Name of
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE		0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER		0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
99995	3837311	1386938868	PAPES	JOAN		0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA		0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA		1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN		1	1	1	0	0	

100000 rows × 65 columns

```
# prompt: remove the provider location column

df.columns
```

```
df.shape
```

```
(100000, 65)
```

```
# prompt: add column as credentials of the provider by concatinating Credentials of the Provider_0', 'Credentials of the Provider_1',
#        'Credentials of the Provider_2', 'Credentials of the Provider_3',
#        'Credentials of the Provider_4', 'Credentials of the Provider_5',
#        'Credentials of the Provider_6', 'Credentials of the Provider_7',
#        'Credentials of the Provider_8,

df['Credentials of the Provider'] = df['Credentials of the Provider_0'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_1'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_2'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_3'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_4'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_5'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_6'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_7'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_8'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_9'].astype(str).fillna('') + ',' + \
                        df['Credentials of the Provider_10'].astype(str).fillna('')
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider_0	Credentials of the Provider_1	Credentials of the Provider_2	Credentials of the Provider_3	Credentials of the Provider_4	.
0	8774979	1891106191	UPADHYAYULA	SATYASREE		0	0	0	0	0	
1	3354385	1346202256	JONES	WENDY	P	0	0	0	0	0	
2	3001884	1306820956	DUROCHER	RICHARD	W	0	0	0	0	0	
3	7594822	1770523540	FULLARD	JASPER		0	0	0	0	0	
4	746159	1073627758	PERROTTI	ANTHONY	E	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...
99995	3837311	1386938868	PAPES	JOAN		0	0	0	0	0	
99996	2079360	1215091327	HAYNER	MARGARET	S	0	0	0	0	0	
99997	8927965	1902868185	VALENCIA	DANA		0	0	0	0	0	
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA		1	1	1	0	0	
99999	3547535	1356772156	RAMEZANI	ELIIAN		1	1	1	0	0	

100000 rows × 66 columns

```
# prompt: drop the Credentials of the Provider_0', 'Credentials of the Provider_1',
# 'Credentials of the Provider_2', 'Credentials of the Provider_3',
# 'Credentials of the Provider_4', 'Credentials of the Provider_5',
# 'Credentials of the Provider_6', 'Credentials of the Provider_7', 'Credentials of the Provider_8', 'Credentials of the Provider_9','Crede

df = df.drop(columns=['Credentials of the Provider_0', 'Credentials of the Provider_1',
'Credentials of the Provider_2', 'Credentials of the Provider_3',
'Credentials of the Provider_4', 'Credentials of the Provider_5',
'Credentials of the Provider_6', 'Credentials of the Provider_7', 'Credentials of the Provider_8', 'Credentials of the Provider_9','Credent
```

```
# prompt: add column city of the provider by concatinating City of the Provider_0', 'City of the Provider_1','City of the Provider_2', 'City

df['City of the Provider'] = df['City of the Provider_0'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_1'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_2'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_3'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_4'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_5'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_6'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_7'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_8'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_9'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_10'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_11'].astype(str).fillna('') + ',' + \
                          df['City of the Provider_12'].astype(str).fillna('')



# prompt: droping the columns  City of the Provider_0', 'City of the Provider_1',
#        'City of the Provider_2', 'City of the Provider_3',
#        'City of the Provider_4', 'City of the Provider_5',
#        'City of the Provider_6', 'City of the Provider_7',
#        'City of the Provider_8', 'City of the Provider_9',
#        'City of the Provider_10', 'City of the Provider_11',
#        'City of the Provider_12

df = df.drop(columns=['City of the Provider_0', 'City of the Provider_1',
       'City of the Provider_2', 'City of the Provider_3',
       'City of the Provider_4', 'City of the Provider_5',
       'City of the Provider_6', 'City of the Provider_7',
       'City of the Provider_8', 'City of the Provider_9',
       'City of the Provider_10', 'City of the Provider_11',
       'City of the Provider_12'])



# prompt: add column state code of the provider by concatinating State Code of the Provider_1', 'State Code of the Provider_2',
#        'State Code of the Provider_3', 'State Code of the Provider_4',
#        'State Code of the Provider_5'

df['State Code of the Provider'] = df['State Code of the Provider_1'].astype(str).fillna('') + ',' + \
                          df['State Code of the Provider_2'].astype(str).fillna('') + ',' + \
                          df['State Code of the Provider_3'].astype(str).fillna('') + ',' + \
                          df['State Code of the Provider_4'].astype(str).fillna('') + ',' + \
                          df['State Code of the Provider_5'].astype(str).fillna('')
df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Gender of the Provider_0	Gender of the Provider_1	Entity Type of the Provider_0	Entity Type of the Provider_1	Street Address 1 of the Provider	...
0	8774979	1891106191	UPADHYAYULA	SATYASREE		0	1	0	1	1402 S GRAND BLVD	...
1	3354385	1346202256	JONES	WENDY	P	0	1	0	1	2950 VILLAGE DR	...
2	3001884	1306820956	DUROCHER	RICHARD	W	1	0	0	1	20 WASHINGTON AVE	...
3	7594822	1770523540	FULLARD	JASPER		1	0	0	1	5746 N BROADWAY ST	...
4	746159	1073627758	PERROTTI	ANTHONY	E	1	0	0	1	875 MILITARY TRL	...
...	...	...	...	...	...	...	...	...	...	...	...
99995	3837311	1386938868	PAPES	JOAN		0	1	0	1	324 E BALTIMORE ST	...
99996	2079360	1215091327	HAYNER	MARGARET	S	0	1	0	1	645 NW 4TH ST	...
99997	8927965	1902868185	VALENCIA	DANA		1	0	0	1	3009 N BALLAS RD	...
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA		0	1	0	1	2365 BOSTON POST RD	...
99999	3547535	1356772156	RAMEZANI	ELIIAN		0	1	0	1	444 MIDDLE NECK RD APT 1H	...

100000 rows × 44 columns

```
# prompt: drop the columns  State Code of the Provider_1', 'State Code of the Provider_2',
#        'State Code of the Provider_3', 'State Code of the Provider_4',
#        'State Code of the Provider_5'

df = df.drop(columns=['State Code of the Provider_1', 'State Code of the Provider_2',
       'State Code of the Provider_3', 'State Code of the Provider_4',
       'State Code of the Provider_5'])



# prompt: add the column country code of the provider by concatinating Country Code of the Provider_0',
#        'Country Code of the Provider_1', 'Country Code of the Provider_2'

df['Country Code of the Provider'] = df['Country Code of the Provider_0'].astype(str).fillna('') + ',' + \
                        df['Country Code of the Provider_1'].astype(str).fillna('') + ',' + \
                        df['Country Code of the Provider_2'].astype(str).fillna('')



# prompt: drop the columns  Country Code of the Provider_0',
#        'Country Code of the Provider_1', 'Country Code of the Provider_2'

df = df.drop(columns=['Country Code of the Provider_0',
       'Country Code of the Provider_1', 'Country Code of the Provider_2'])
```

```
# prompt: add the column provider type by concatinating  Provider Type_0', 'Provider Type_1', 'Provider Type_2',
#        'Provider Type_3', 'Provider Type_4', 'Provider Type_5',and after adding drop Provider Type_0', 'Provider Type_1', 'Provider Type_
#        'Provider Type_3', 'Provider Type_4', 'Provider Type_5',
#        'Provider Type_6'
#        'Provider Type_6'

df['Provider Type'] = df['Provider Type_0'].astype(str).fillna('') + ',' + \
                        df['Provider Type_1'].astype(str).fillna('') + ',' + \
                        df['Provider Type_2'].astype(str).fillna('') + ',' + \
                        df['Provider Type_3'].astype(str).fillna('') + ',' + \
                        df['Provider Type_4'].astype(str).fillna('') + ',' + \
                        df['Provider Type_5'].astype(str).fillna('') + ',' + \
                        df['Provider Type_6'].astype(str).fillna('')

df = df.drop(columns=['Provider Type_0', 'Provider Type_1', 'Provider Type_2',
       'Provider Type_3', 'Provider Type_4', 'Provider Type_5',
       'Provider Type_6'])

df
```

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Gender of the Provider_0	Gender of the Provider_1	Entity Type of the Provider_0	Entity Type of the Provider_1	Street Address 1 of the Provider	...
0	8774979	1891106191	UPADHYAYULA	SATYASREE		0	1	0	1	1402 S GRAND BLVD	...
1	3354385	1346202256	JONES	WENDY	P	0	1	0	1	2950 VILLAGE DR	...
2	3001884	1306820956	DUROCHER	RICHARD	W	1	0	0	1	20 WASHINGTON AVE	...
3	7594822	1770523540	FULLARD	JASPER		1	0	0	1	5746 N BROADWAY ST	...
4	746159	1073627758	PERROTTI	ANTHONY	E	1	0	0	1	875 MILITARY TRL	...
...	...	...	...	...	...	...	...	...	...	...	...
99995	3837311	1386938868	PAPES	JOAN		0	1	0	1	324 E BALTIMORE ST	...
99996	2079360	1215091327	HAYNER	MARGARET	S	0	1	0	1	645 NW 4TH ST	...
99997	8927965	1902868185	VALENCIA	DANA		1	0	0	1	3009 N BALLAS RD	...
99998	8854571	1891941183	GONZALEZLAMOS	RAFAELA		0	1	0	1	2365 BOSTON POST RD	...
99999	3547535	1356772156	RAMEZANI	ELIIAN		0	1	0	1	444 MIDDLE NECK RD APT 1H	...

100000 rows × 34 columns

```
# prompt: drop the columns  index', 'National Provider Identifier',
#        'Last Name/Organization Name of the Provider',
#        'First Name of the Provider', 'Middle Initial of the Provider',Zip Code of the Provider

df.drop(columns=['index', 'National Provider Identifier',
       'Last Name/Organization Name of the Provider',
       'First Name of the Provider', 'Middle Initial of the Provider','Zip Code of the Provider'], inplace=True)
df
```

	Gender of the Provider_0	Gender of the Provider_1	Entity Type of the Provider_0	Entity Type of the Provider_1	Street Address 1 of the Provider	Street Address 2 of the Provider	State Code of the Provider_0	Country Code of the Provider_0	Country Code of the Provider_1	Country Code of the Provider_2	...	Ave Medi All Am
0	0	1	0	1	1402 S GRAND BLVD	FDT 14TH FLOOR	0	0	0	1	...	0.20
1	0	1	0	1	2950 VILLAGE DR	NaN	0	0	0	1	...	0.12
2	1	0	0	1	20 WASHINGTON AVE	STE 212	0	0	0	1	...	0.09
3	1	0	0	1	5746 N BROADWAY ST	NaN	0	0	0	1	...	0.00
4	1	0	0	1	875 MILITARY TRL	SUITE 200	0	0	0	1	...	0.02
...	...	...	...	...	...	...	...	...	...	...	...	
99995	0	1	0	1	324 E BALTIMORE ST	NaN	0	0	0	1	...	0.08
99996	0	1	0	1	645 NW 4TH ST	NaN	0	0	0	1	...	0.06
99997	1	0	0	1	3009 N BALLAS RD	SUITE 202B	0	0	0	1	...	0.01
99998	0	1	0	1	2365 BOSTON POST RD	SUITE 201	0	0	0	1	...	0.03
99999	0	1	0	1	444 MIDDLE NECK RD APT 1H	NaN	0	0	0	1	...	0.03

100000 rows × 28 columns

```
# prompt: concatinate Gender of the Provider_0', 'Gender of the Provider_1 as Gender of the provider and reomove Gender of the Provider_0',

df['Gender of the Provider'] = df['Gender of the Provider_0'].astype(str).fillna('') + ',' + \
                       df['Gender of the Provider_1'].astype(str).fillna('')
df = df.drop(columns=['Gender of the Provider_0', 'Gender of the Provider_1'])

df['Entity Type of the Provider'] = df['Entity Type of the Provider_0'].astype(str).fillna('') + ',' + \
                       df['Entity Type of the Provider_1'].astype(str).fillna('')
df = df.drop(columns=['Entity Type of the Provider_0', 'Entity Type of the Provider_1'])

df
```

	Street Address 1 of the Provider	Street Address 2 of the Provider	State Code of the Provider_0	Country Code of the Provider_0	Country Code of the Provider_1	Country Code of the Provider_2	Medicare Participation Indicator	Place of Service	HCPCS Code	HCPCS Description	...	Avera Medica Payme Amou
0	1402 S GRAND BLVD	FDT 14TH FLOOR	0	0	0	1	Y	F	99223	Initial hospital inpatient care typically 70 m...	...	0.1574
1	2950 VILLAGE DR	NaN	0	0	0	1	Y	O	G0202	Screening mammography bilateral 2view study of...	...	0.1189
2	20 WASHINGTON AVE	STE 212	0	0	0	1	Y	O	99348	Established patient home visit typically 25 mi...	...	0.0645
3	5746 N BROADWAY ST	NaN	0	0	0	1	Y	O	81002	Urinalysis manual test	...	0.0034
4	875 MILITARY TRL	SUITE 200	0	0	0	1	Y	O	96372	Injection beneath the skin or into muscle for ...	...	0.0195
...	...	...	...	...	...	...	...	...	...	...	...	
99995	324 E BALTIMORE ST	NaN	0	0	0	1	Y	O	97162	Evaluation of physical therapy typically 30 mi...	...	0.0608
99996	645 NW 4TH ST	NaN	0	0	0	1	Y	O	99213	Established patient office or other outpatient...	...	0.0300
99997	3009 N BALLAS RD	SUITE 202B	0	0	0	1	Y	F	93320	Doppler ultrasound study of heart blood flow v...	...	0.0141
99998	2365 BOSTON POST RD	SUITE 201	0	0	0	1	Y	O	G0008	Administration of influenza virus vaccine	...	0.0299
99999	444 MIDDLE NECK RD APT 1H	NaN	0	0	0	1	Y	O	97112	Therapeutic procedure to reeducate braintonerv...	...	0.0297

100000 rows × 26 columns

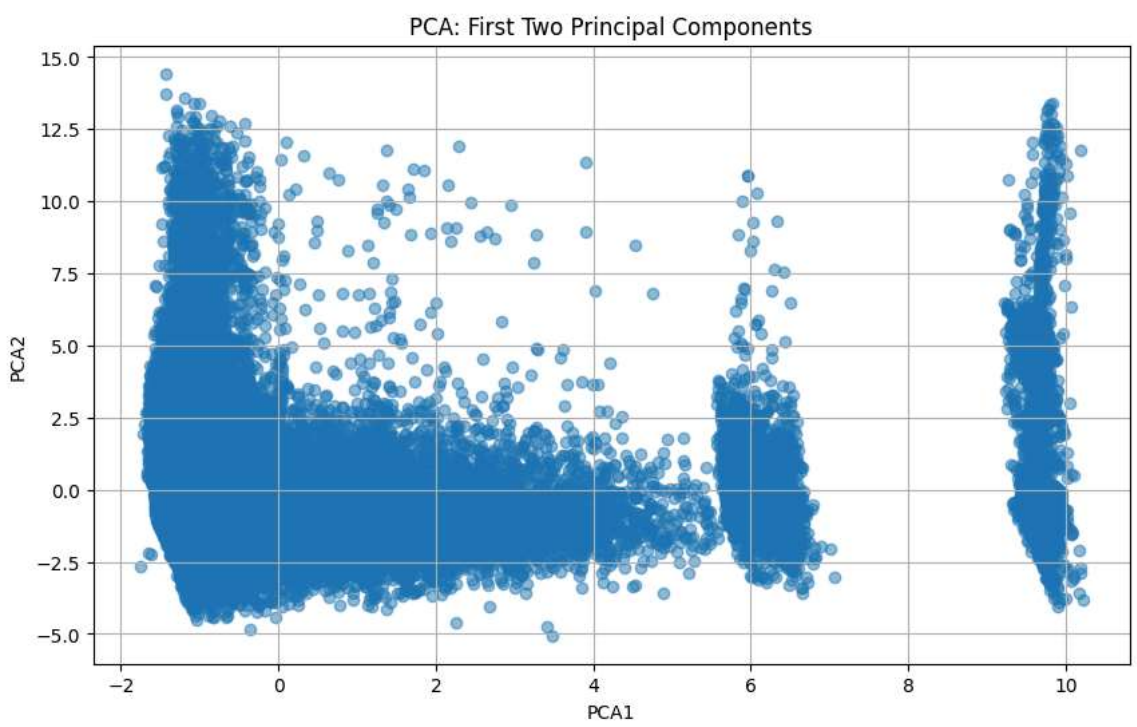
```
# prompt: concatenate   State Code of the Provider with  State Code of the Provider_0 and assign it to  State Code of the Provider

df['State Code of the Provider'] = df['State Code of the Provider_0'].astype(str).fillna('') + ',' + df['State Code of the Provider']
df = df.drop(columns=['State Code of the Provider_0'])
df
```

	Street Address 1 of the Provider	Street Address 2 of the Provider	Country Code of the Provider_0	Country Code of the Provider_1	Country Code of the Provider_2	Medicare Participation Indicator	Place of Service	HCPCS Code	HCPCS Description	HCPCS Drug Indicator	...	Averag Medicar Paymen Amoun
0	1402 S GRAND BLVD	FDT 14TH FLOOR	0	0	1	Y	F	99223	Initial hospital inpatient care typically 70 m...	N	...	0.15747
1	2950 VILLAGE DR	NaN	0	0	1	Y	O	G0202	Screening mammography bilateral 2view study of...	N	...	0.11898
2	20 WASHINGTON AVE	STE 212	0	0	1	Y	O	99348	Established patient home visit typically 25 mi...	N	...	0.06452
3	5746 N BROADWAY ST	NaN	0	0	1	Y	O	81002	Urinalysis manual test	N	...	0.00342
4	875 MILITARY TRL	SUITE 200	0	0	1	Y	O	96372	Injection beneath the skin or into muscle for ...	N	...	0.01955
...	...	...	...	...	...	...	...	...	...	...	...	.
99995	324 E BALTIMORE ST	NaN	0	0	1	Y	O	97162	Evaluation of physical therapy typically 30 mi...	N	...	0.06080
99996	645 NW 4TH ST	NaN	0	0	1	Y	O	99213	Established patient office or other outpatient...	N	...	0.03003
99997	3009 N BALLAS RD	SUITE 202B	0	0	1	Y	F	93320	Doppler ultrasound study of heart blood flow v...	N	...	0.01410
99998	2365 BOSTON POST RD	SUITE 201	0	0	1	Y	O	G0008	Administration of influenza virus vaccine	N	...	0.02996
99999	444 MIDDLE NECK RD APT 1H	NaN	0	0	1	Y	O	97112	Therapeutic procedure to reeducate braintonerv...	N	...	0.02977

100000 rows × 25 columns

PCA of the first two PCA components

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Load the dataset
file_path = '/content/_preprocessed and cleaned healthcare providers (1).csv'
df = pd.read_csv(file_path)

# Preprocess the data: replace 'path_to_your_file.csv' with your file path
df.fillna(0, inplace=True)  # Handle missing values

# Select numeric columns for PCA
numeric_df = df.select_dtypes(include=[float, int])

# Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numeric_df)

# Apply PCA to reduce to the first two principal components
pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_data)

# Create a dataframe with the PCA components
pca_df = pd.DataFrame(data=pca_components, columns=['PCA1', 'PCA2'])

# Scatter plot of the first two PCA components
plt.figure(figsize=(10, 6))
plt.scatter(pca_df['PCA1'], pca_df['PCA2'], alpha=0.5)
plt.title('PCA: First Two Principal Components')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.grid(True)
plt.show()
```

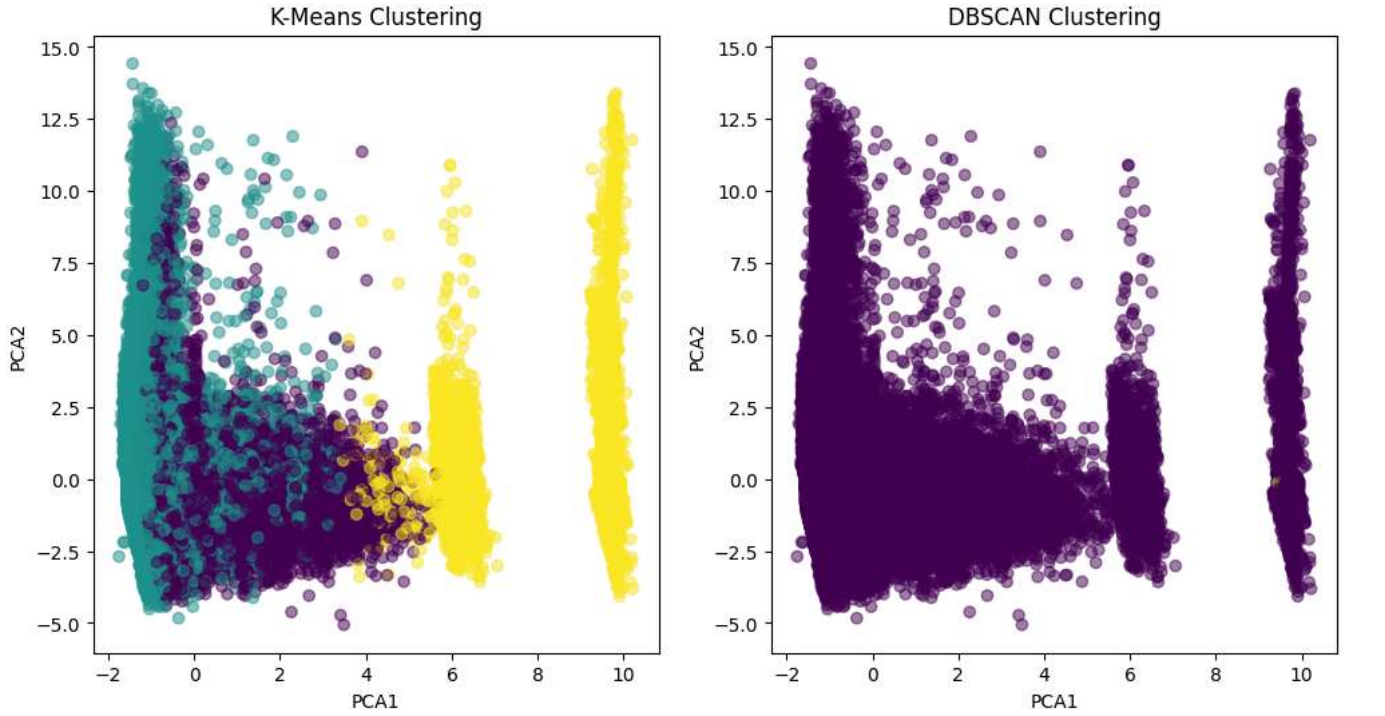


plot of the first two PCA components with DBSCAN labels.

> plot of the first two PCA components with K-Means labels

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, DBSCAN
import matplotlib.pyplot as plt

# Load the dataset
file_path = '/content/_preprocessed and cleaned healthcare providers (1).csv'
df = pd.read_csv(file_path)

# Preprocess the data: replace missing values with 0
df.fillna(0, inplace=True)

# Select numeric columns for clustering
numeric_df = df.select_dtypes(include=[float, int])

# Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numeric_df)

# Apply K-Means
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(scaled_data)

# Apply DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(scaled_data)

# Apply PCA to reduce to the first two principal components
pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_data)

# Create a dataframe with the PCA components and cluster labels
pca_df = pd.DataFrame(data=pca_components, columns=['PCA1', 'PCA2'])
pca_df['kmeans_labels'] = kmeans_labels
pca_df['dbscan_labels'] = dbscan_labels

# Scatter plot of the first two PCA components with K-Means labels
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['kmeans_labels'], cmap='viridis', alpha=0.5)
plt.title('K-Means Clustering')
plt.xlabel('PCA1')
plt.ylabel('PCA2')

# Scatter plot of the first two PCA components with DBSCAN labels
plt.subplot(1, 2, 2)
plt.scatter(pca_df['PCA1'], pca_df['PCA2'], c=pca_df['dbscan_labels'], cmap='viridis', alpha=0.5)
plt.title('DBSCAN Clustering')
plt.xlabel('PCA1')
plt.ylabel('PCA2')

plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
```



Showing the number of data points in each cluster by k-means

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Selecting relevant numerical columns for k-means clustering
numerical_columns = [
    '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'
]

# Extract the data for these columns
# Use 'data_for_clustering' instead of 'data'
data_for_clustering = data_for_clustering[numerical_columns]

# Handle any missing values by filling them with the mean of the column
data_for_clustering.fillna(data_for_clustering.mean(), inplace=True)

# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_for_clustering)

# Perform k-means clustering with a chosen number of clusters, e.g., 3
kmeans = KMeans(n_clusters=3, random_state=42)
# You'll likely want to assign the clusters to the original dataframe,
# so let's assume you have a dataframe named 'df' where you want to store the cluster assignments.
# If you don't have 'df' defined, replace it with the appropriate dataframe name.
df['Cluster'] = kmeans.fit_predict(data_scaled)

# Show the number of data points in each cluster
df['Cluster'].value_counts()
```

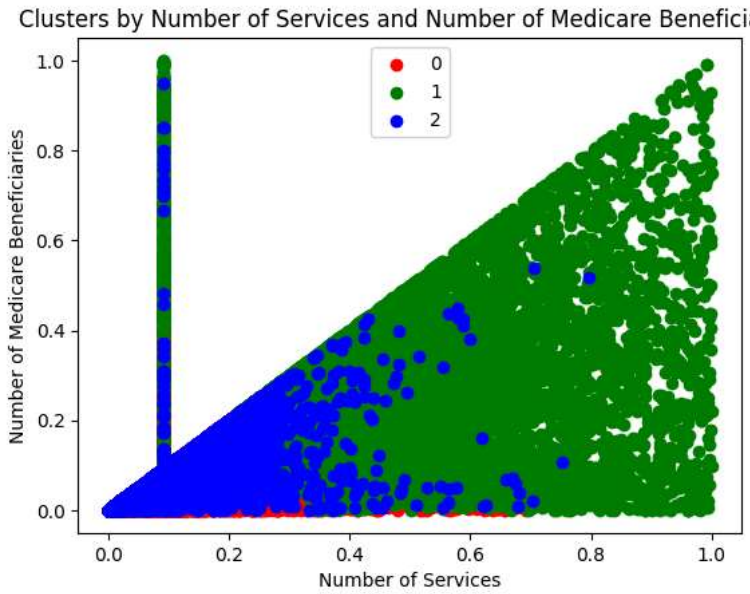
```
<ipython-input-37-279997af9916>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  data_for_clustering.fillna(data_for_clustering.mean(), inplace=True)
```

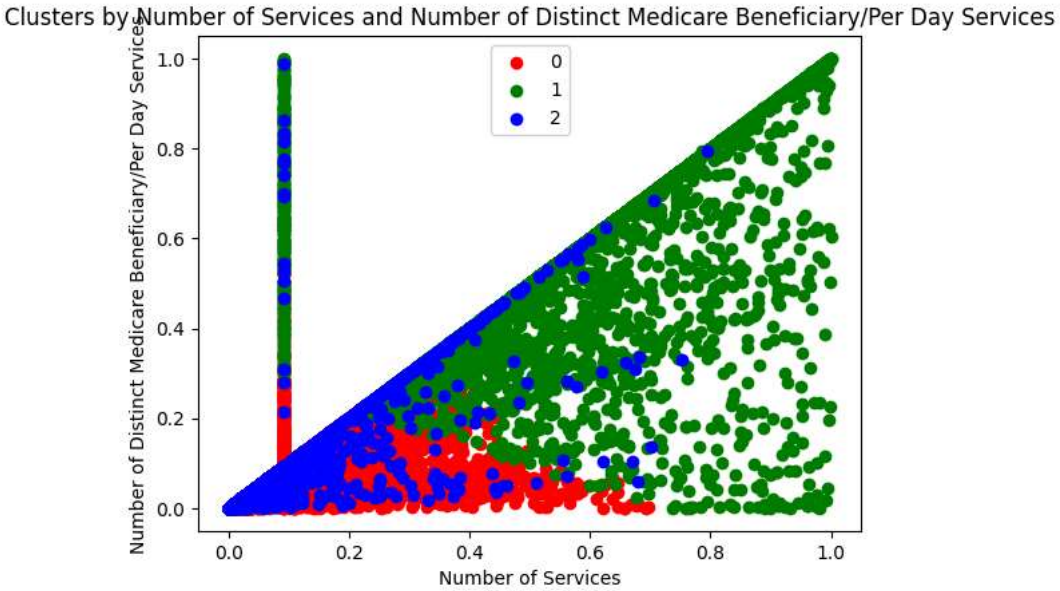
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
Cluster
0    74428
2    15551
1    10021
Name: count, dtype: int64
```

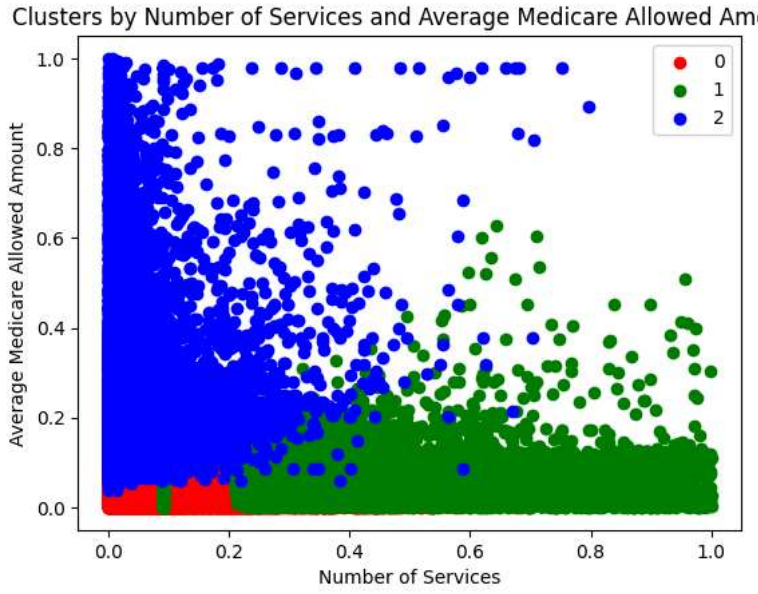
Adding visualizations of clusters created above using scatter plots

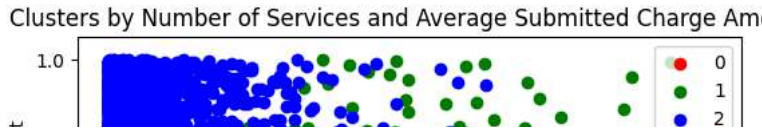
```
# prompt: Add visualizations of clusters created above  using scatter plots.

import matplotlib.pyplot as plt

# Define colors for different clusters
colors = ['red', 'green', 'blue']

# Create a scatter plot for each pair of features
for i in range(len(numerical_columns)):
    for j in range(i + 1, len(numerical_columns)):
        plt.figure()
        for cluster, color in zip(range(3), colors):
            subset = df[df['Cluster'] == cluster]
            plt.scatter(subset[numerical_columns[i]], subset[numerical_columns[j]], c=color, label=cluster)
        plt.xlabel(numerical_columns[i])
        plt.ylabel(numerical_columns[j])
        plt.title('Clusters by {} and {}'.format(numerical_columns[i], numerical_columns[j]))
        plt.legend()
        plt.show()
```

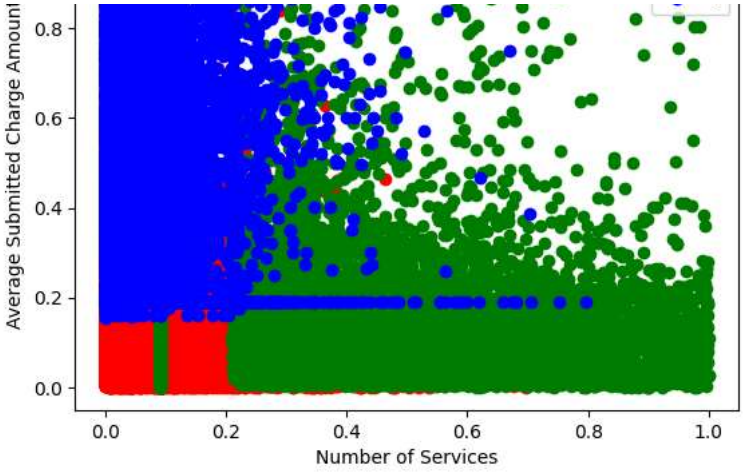
[↓]


Clusters by Number of Services and Number of Medicare Beneficiaries
Number of Medicare Beneficiaries
Number of Services
0
1
2

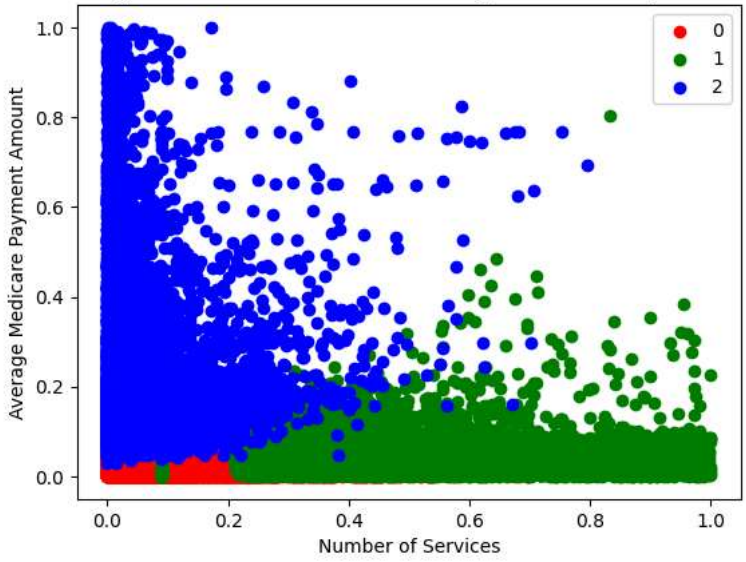

Clusters by Number of Services and Number of Distinct Medicare Beneficiary/Per Day Services
Number of Distinct Medicare Beneficiary/Per Day Services
Number of Services
0
1
2


Clusters by Number of Services and Average Medicare Allowed Amount
Average Medicare Allowed Amount
Number of Services
0
1
2

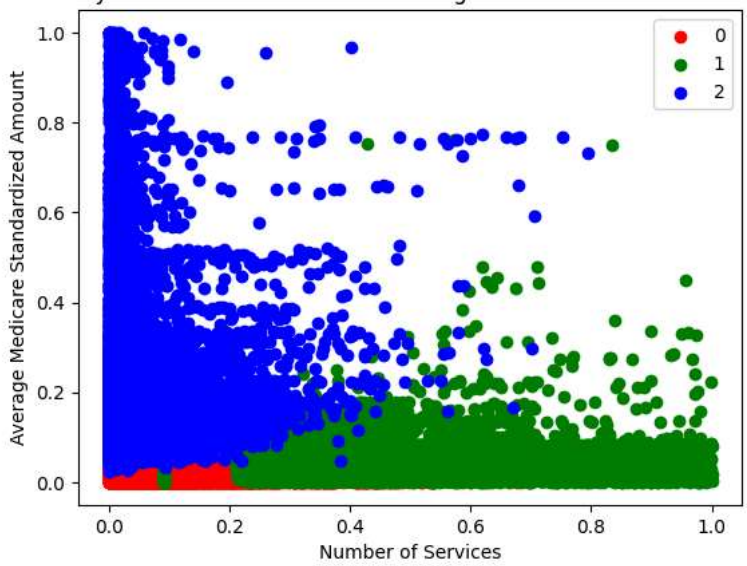

Clusters by Number of Services and Average Submitted Charge Amount
0
1
2



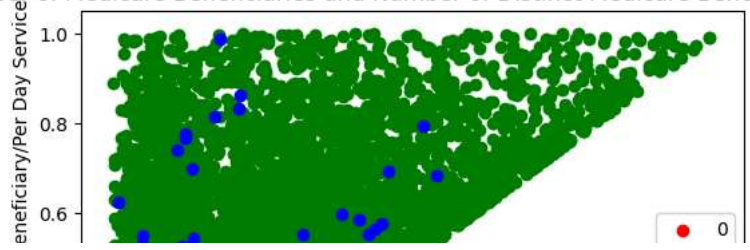
Clusters by Number of Services and Average Medicare Payment Amount

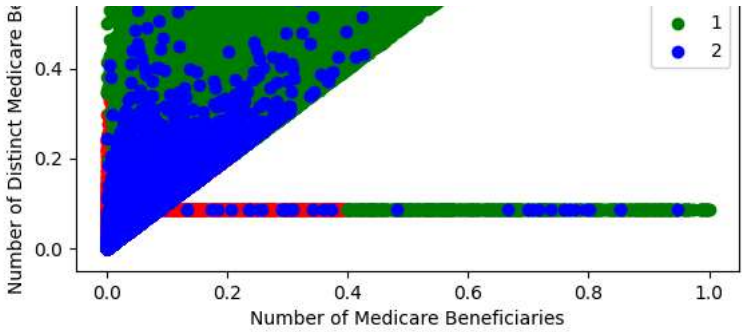


Clusters by Number of Services and Average Medicare Standardized Amount

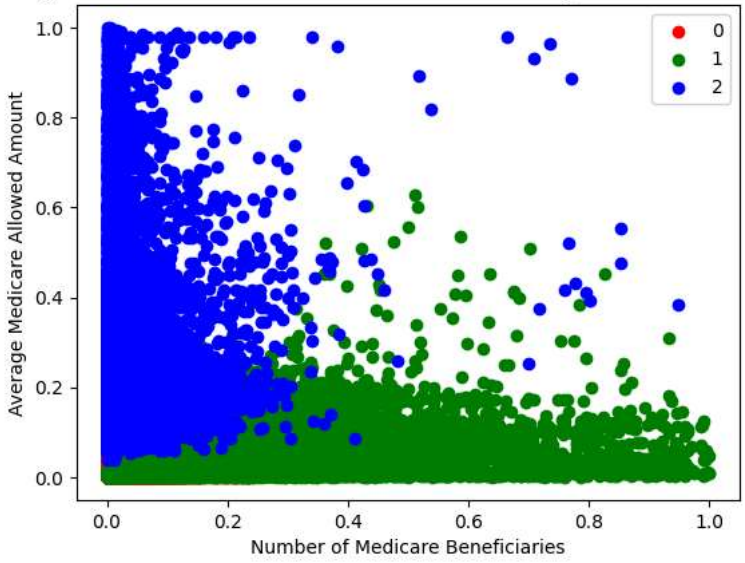


Clusters by Number of Medicare Beneficiaries and Number of Distinct Medicare Beneficiary/Per Day Services

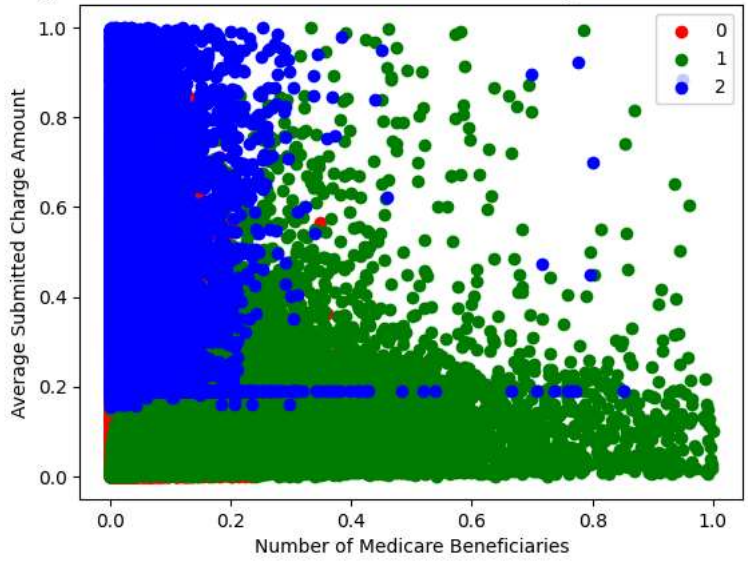




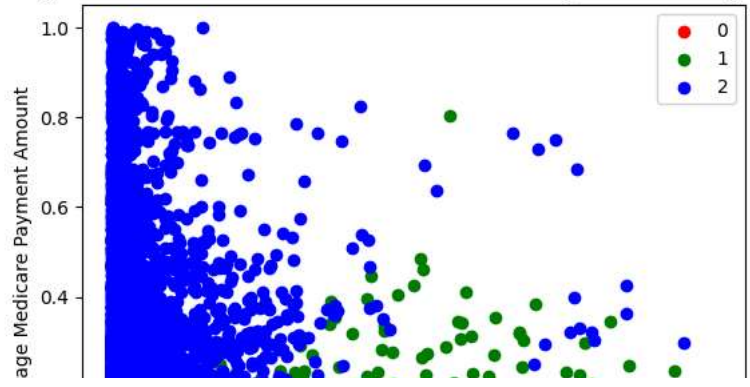
Clusters by Number of Medicare Beneficiaries and Average Medicare Allowed Amount

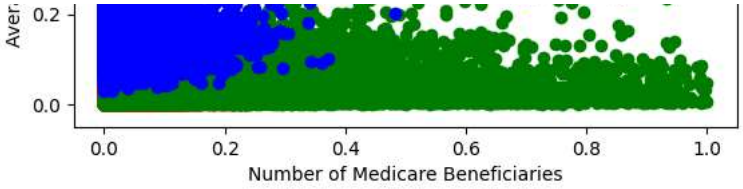


Clusters by Number of Medicare Beneficiaries and Average Submitted Charge Amount

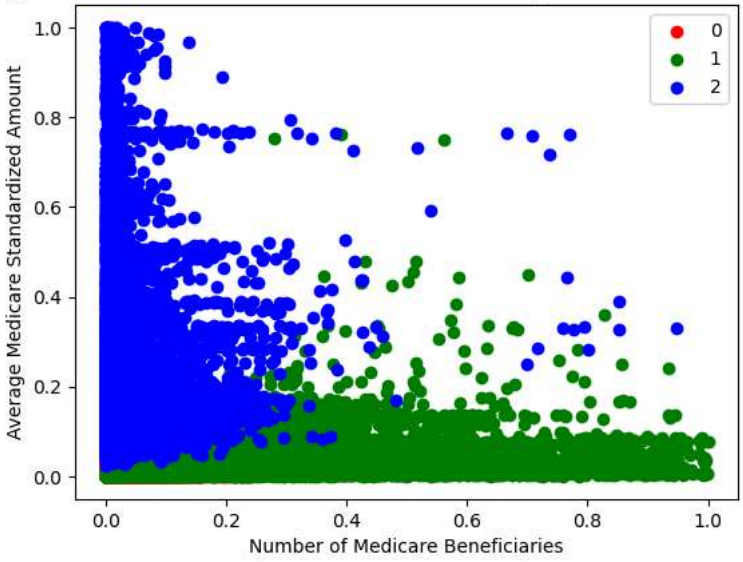


Clusters by Number of Medicare Beneficiaries and Average Medicare Payment Amount

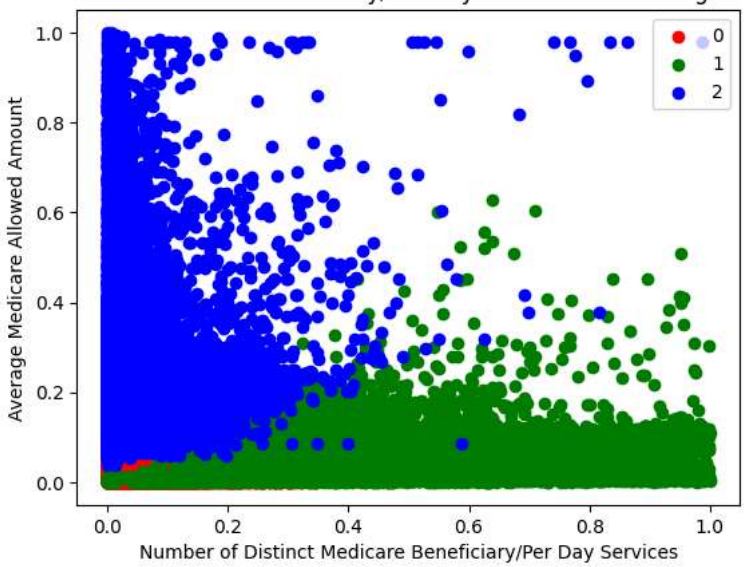




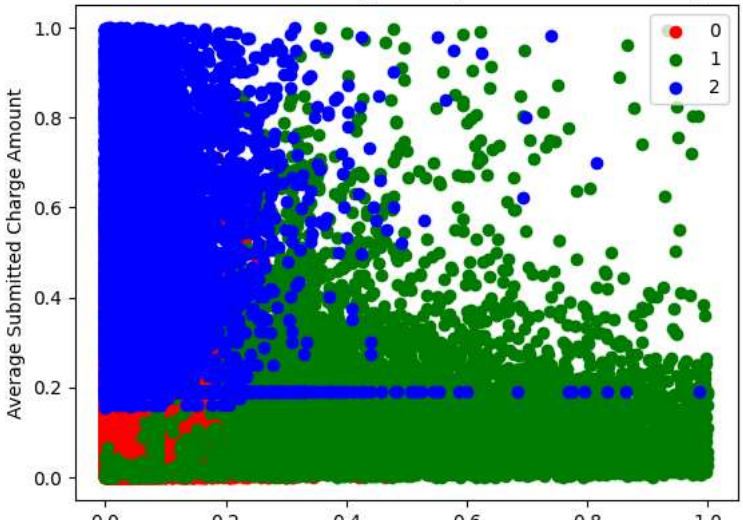
Clusters by Number of Medicare Beneficiaries and Average Medicare Standardized Amount



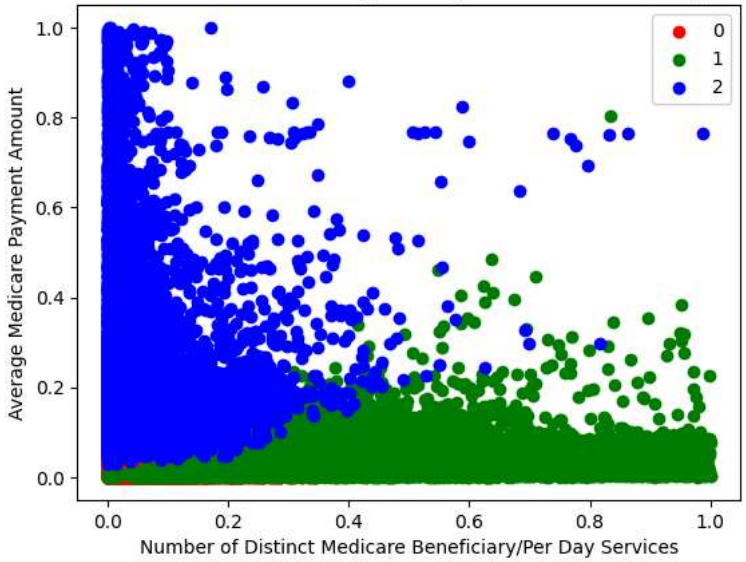
Clusters by Number of Distinct Medicare Beneficiary/Per Day Services and Average Medicare Allowed Amount



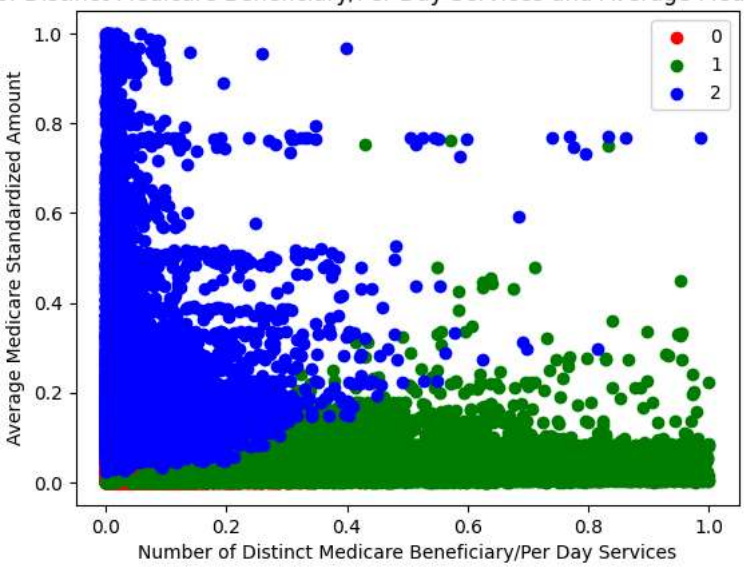
Clusters by Number of Distinct Medicare Beneficiary/Per Day Services and Average Submitted Charge Amount



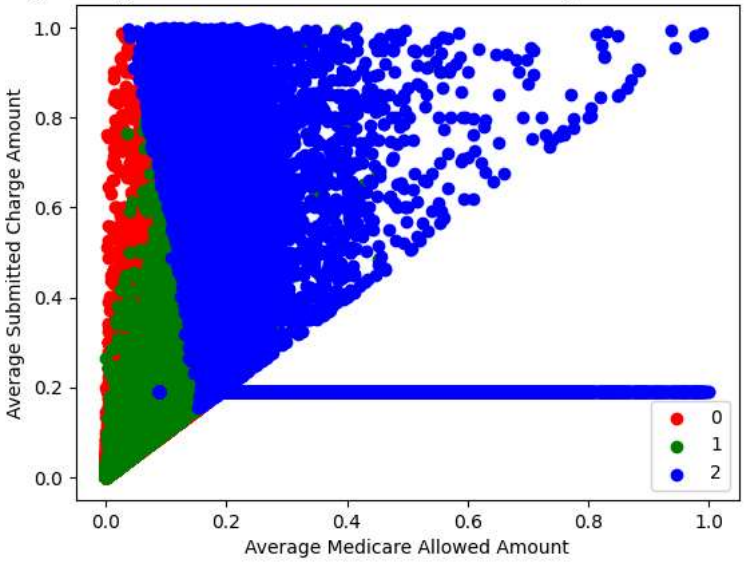
Clusters by Number of Distinct Medicare Beneficiary/Per Day Services and Average Medicare Payment Amount



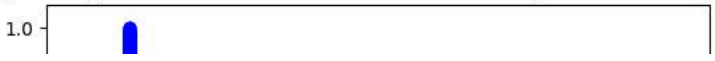
Clusters by Number of Distinct Medicare Beneficiary/Per Day Services and Average Medicare Standardized Amount



Clusters by Average Medicare Allowed Amount and Average Submitted Charge Amount



Clusters by Average Medicare Allowed Amount and Average Medicare Payment Amount



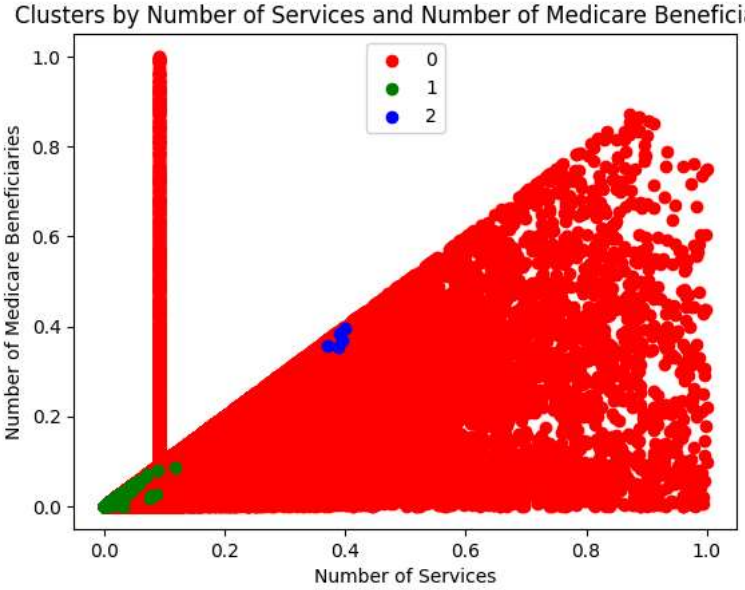
Start coding or generate with AI.

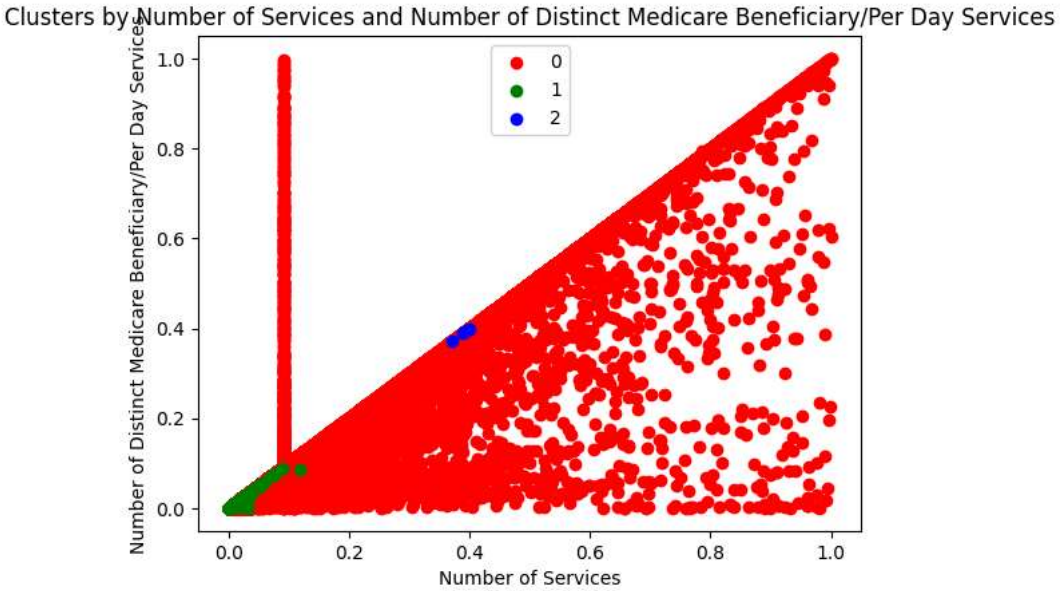
```
from sklearn.cluster import DBSCAN

# Perform DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)  # Example parameters, these may need tuning
data_for_clustering['DBSCAN_Cluster'] = dbscan.fit_predict(data_scaled) # Use 'data_for_clustering'

# Show the number of data points in each cluster
data_for_clustering['DBSCAN_Cluster'].value_counts()
```

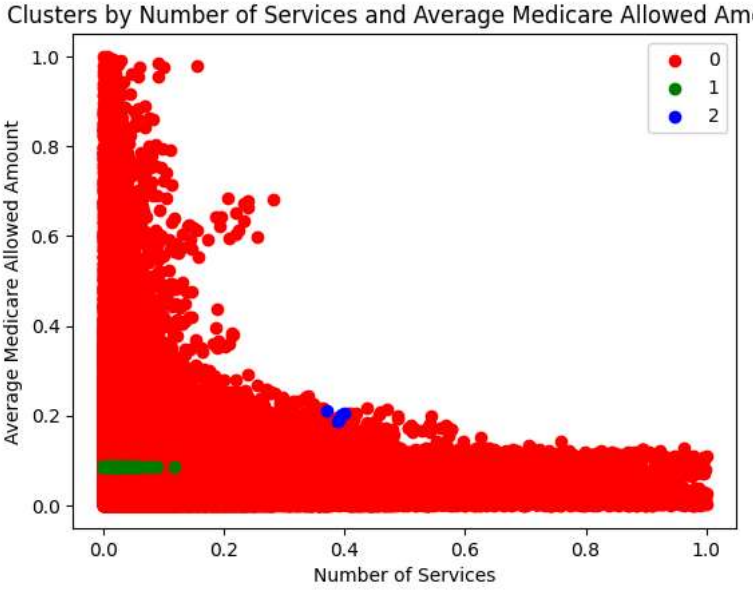
```
DBSCAN_Cluster
 0     97984
-1      1573
 1       226
 4        26
 5        21
 9        13
 7        13
 10       12
 12       11
 3        10
 6         8
 23        8
 16        7
 17        5
 19        5
 22        5
 21        5
 20        5
 29        5
 18        5
 30        5
 15        5
 13        5
 2         5
 8         5
 14        5
 27        5
 24        4
 28        4
 11        4
 26        4
 25        2
Name: count, dtype: int64
```

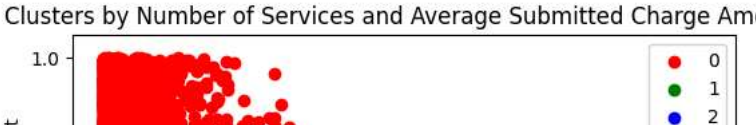
Start coding or generate with AI.

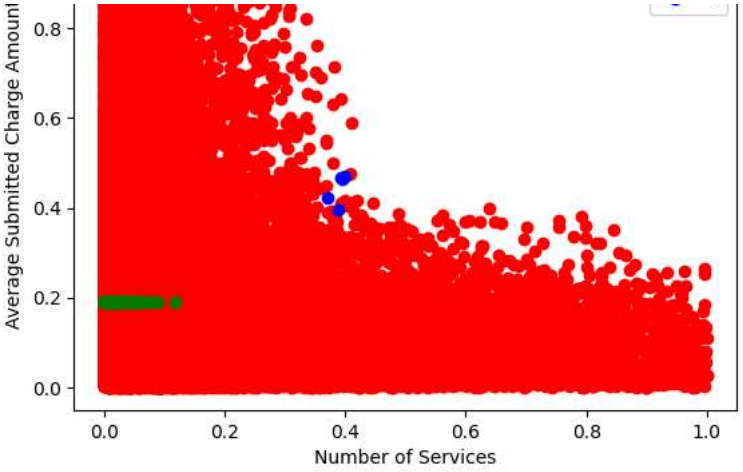
```
import matplotlib.pyplot as plt # Correct the import statement

# Define colors for different clusters
colors = ['red', 'green', 'blue']

# Create a scatter plot for each pair of features
for i in range(len(numerical_columns)):
    for j in range(i + 1, len(numerical_columns)):
        plt.figure()
        for cluster, color in zip(range(3), colors):
            subset = data_for_clustering[data_for_clustering['DBSCAN_Cluster'] == cluster]
            plt.scatter(subset[numerical_columns[i]], subset[numerical_columns[j]], c=color, label=cluster)
        plt.xlabel(numerical_columns[i])
        plt.ylabel(numerical_columns[j])
        plt.title('Clusters by {} and {}'.format(numerical_columns[i], numerical_columns[j]))
        plt.legend()
        plt.show()
```


Clusters by Number of Services and Number of Medicare Beneficiaries
Number of Medicare Beneficiaries
Number of Services
0
1
2

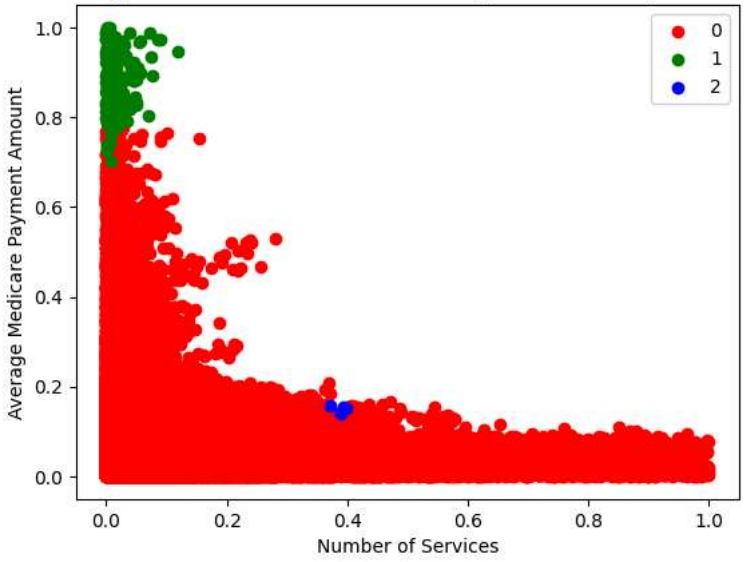

Clusters by Number of Services and Number of Distinct Medicare Beneficiary/Per Day Services
Number of Distinct Medicare Beneficiary/Per Day Services
Number of Services
0
1
2


Clusters by Number of Services and Average Medicare Allowed Amount
Average Medicare Allowed Amount
Number of Services
0
1
2

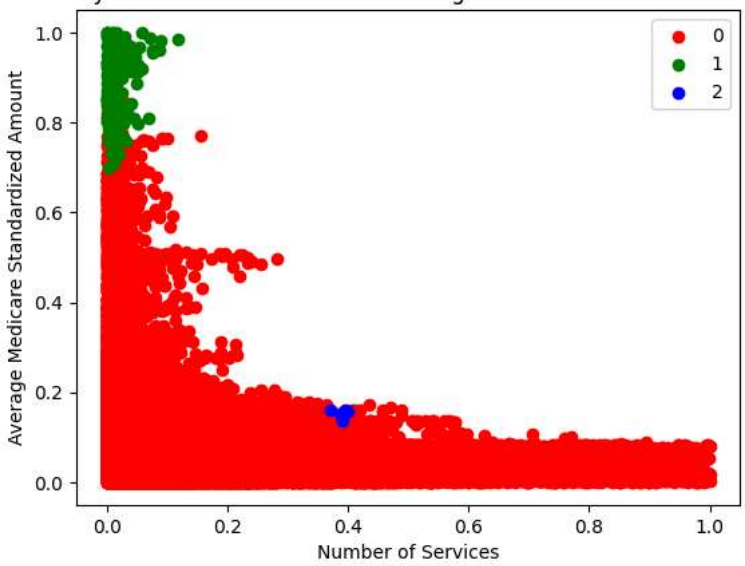

Clusters by Number of Services and Average Submitted Charge Amount
0
1
2



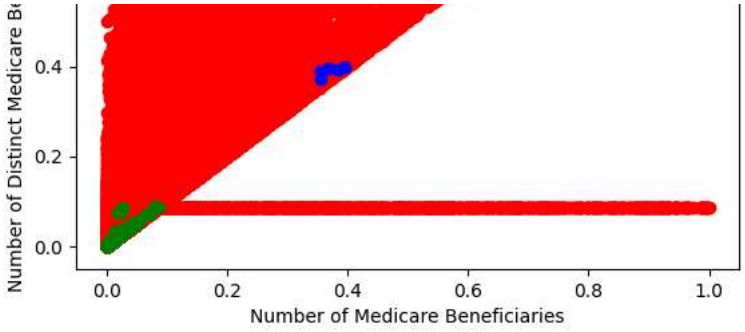
Clusters by Number of Services and Average Medicare Payment Amount



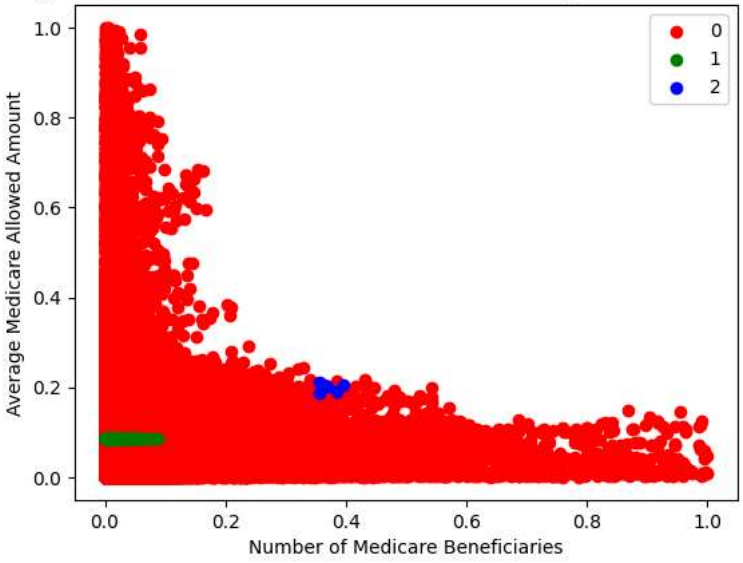
Clusters by Number of Services and Average Medicare Standardized Amount



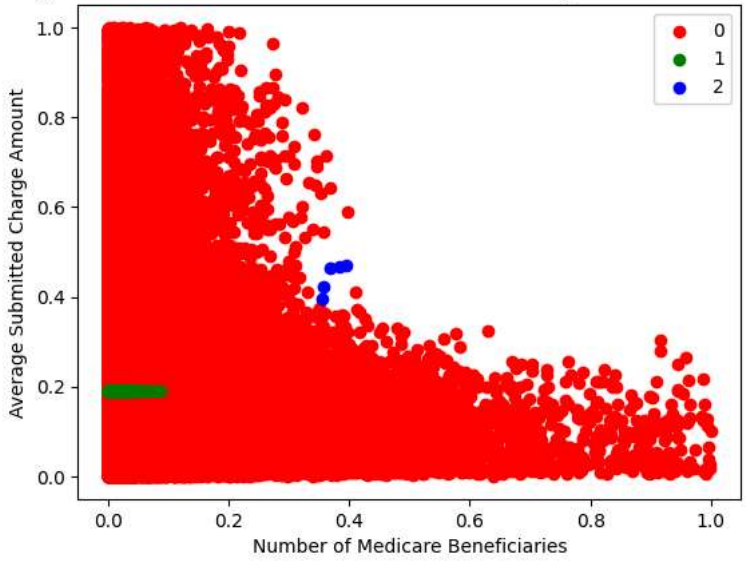
Clusters by Number of Medicare Beneficiaries and Number of Distinct Medicare Beneficiary/Per Day Services



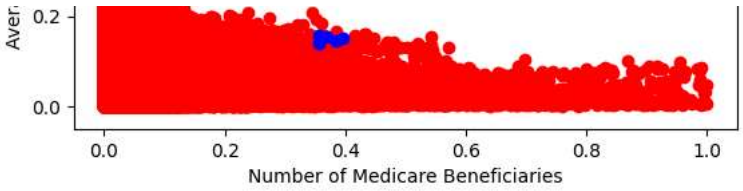
Clusters by Number of Medicare Beneficiaries and Average Medicare Allowed Amount



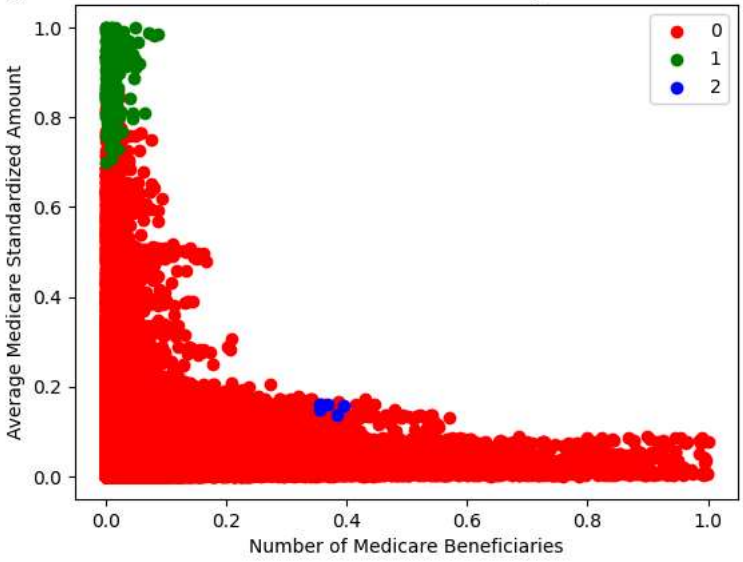
Clusters by Number of Medicare Beneficiaries and Average Submitted Charge Amount



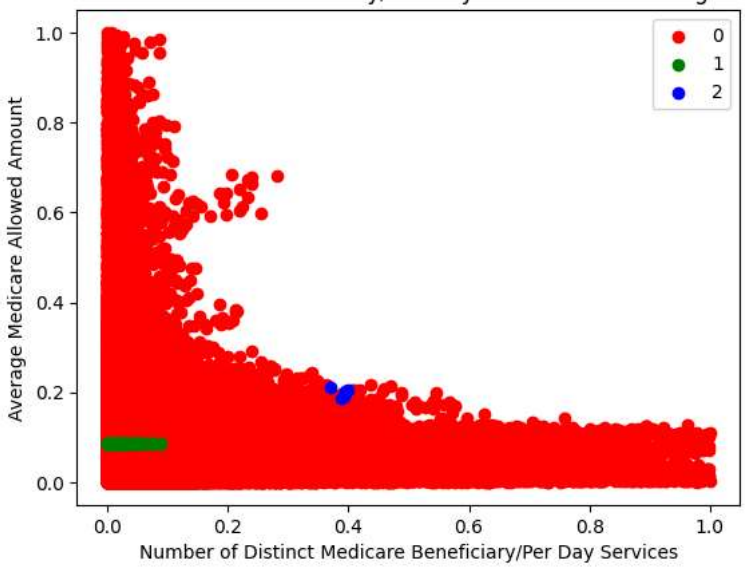
Clusters by Number of Medicare Beneficiaries and Average Medicare Payment Amount



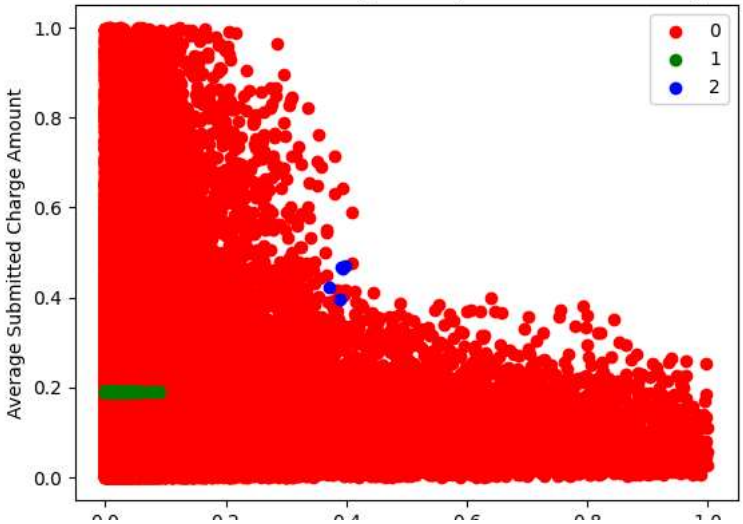
Clusters by Number of Medicare Beneficiaries and Average Medicare Standardized Amount



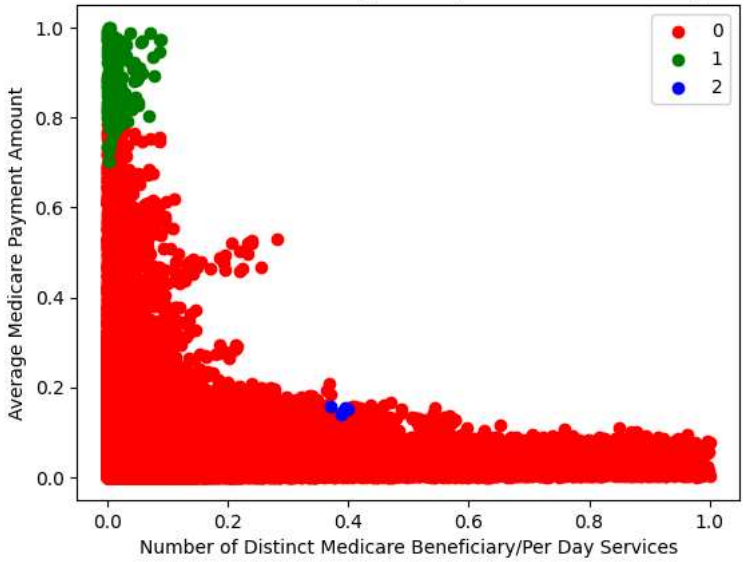
Clusters by Number of Distinct Medicare Beneficiary/Per Day Services and Average Medicare Allowed Amount



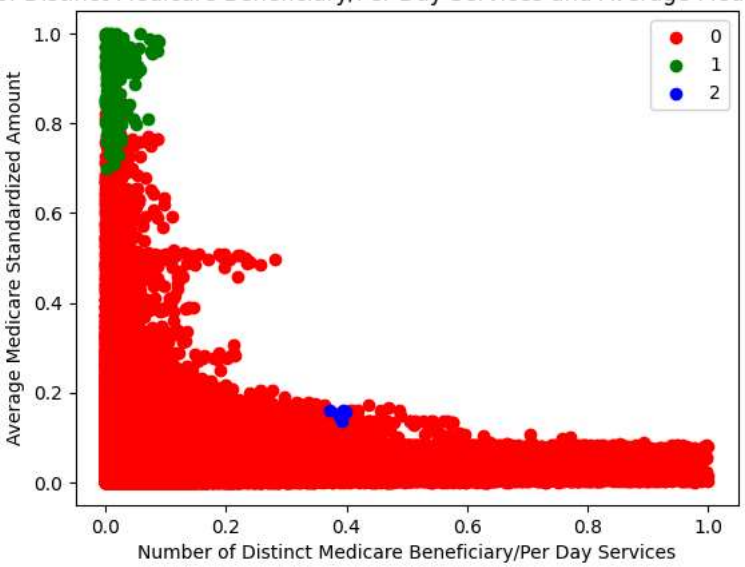
Clusters by Number of Distinct Medicare Beneficiary/Per Day Services and Average Submitted Charge Amount



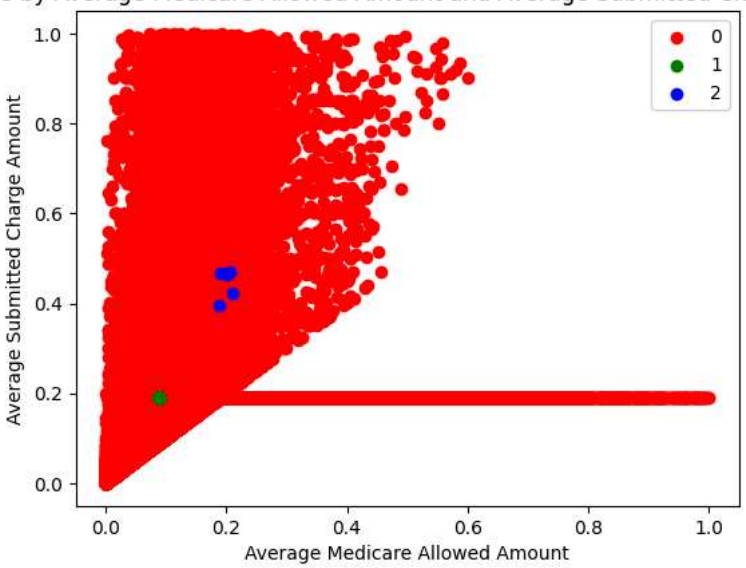
Clusters by Number of Distinct Medicare Beneficiary/Per Day Services and Average Medicare Payment Amount



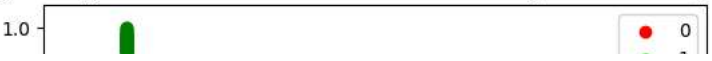
Clusters by Number of Distinct Medicare Beneficiary/Per Day Services and Average Medicare Standardized Amount

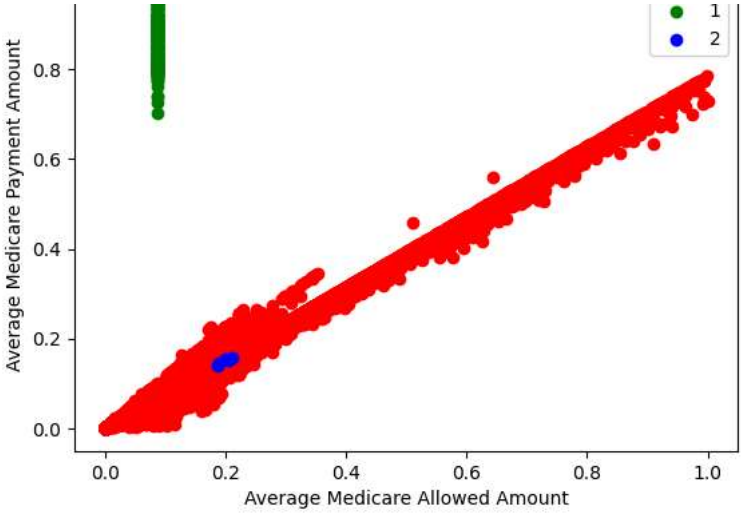


Clusters by Average Medicare Allowed Amount and Average Submitted Charge Amount

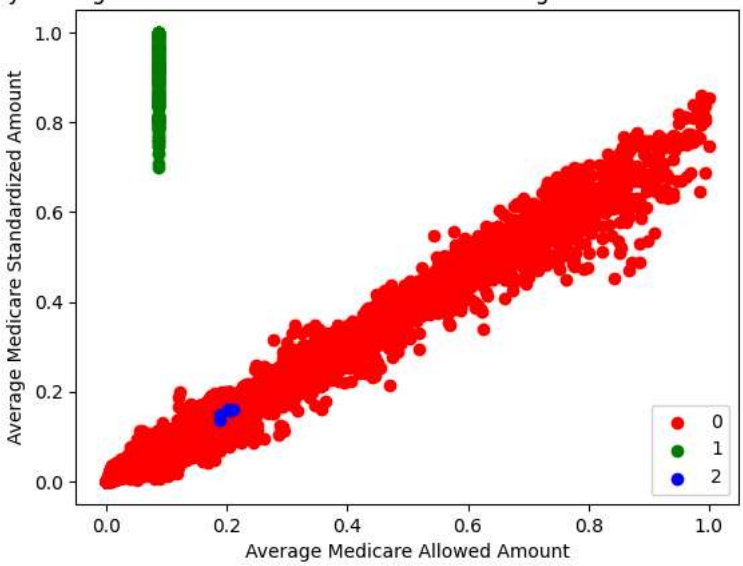


Clusters by Average Medicare Allowed Amount and Average Medicare Payment Amount

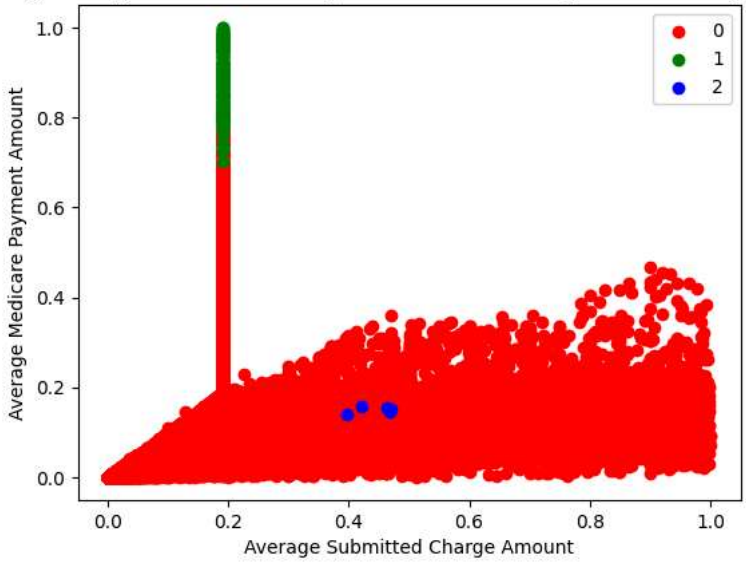




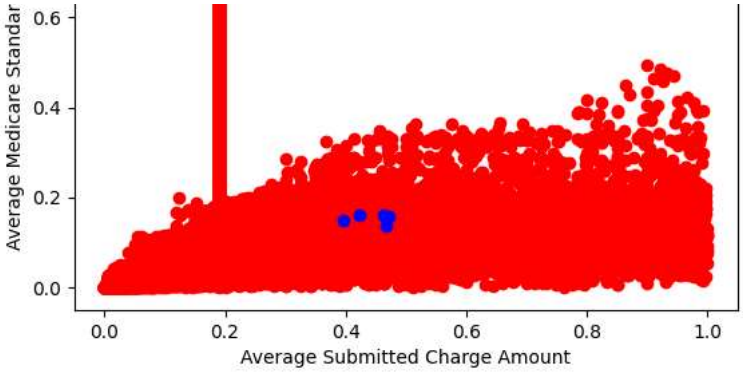
Clusters by Average Medicare Allowed Amount and Average Medicare Standardized Amount



Clusters by Average Submitted Charge Amount and Average Medicare Payment Amount



Clusters by Average Submitted Charge Amount and Average Medicare Standardized Amount



Clusters by Average Medicare Payment Amount and Average Medicare Standardized Amount