JWillis_D209_Data_Mining_PA1

January 8, 2023

0.1 D209 - Data Mining I - PA1

0.1.1 Import Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from pandas import DataFrame
import sklearn.neighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
```

0.1.2 Load Data From medical_clean.csv

[2]:
```
# load data file
df = pd.read_csv('medical_clean.csv')
# quick test the data is present and see the shape
df.head()
```

[2]:
```
   CaseOrder Customer_id                           Interaction  \
0          1     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
1          2     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
2          3     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
3          4     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
4          5     C544523  5885f56b-d6da-43a3-8760-83583af94266

                                UID          City State        County    Zip  \
0  3a83ddb66e2ae73798bdf1d705dc0932           Eva    AL        Morgan  35621
1  176354c5eef714957d486009feabf195      Marianna    FL       Jackson  32446
2  e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD     Minnehaha  57110
3  cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca  56072
4  d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William  23181

        Lat        Lng  …  TotalCharge Additional_charges Item1 Item2  Item3  \
```

```
0  34.34960 -86.72508  …  3726.702860       17939.403420     3     3       2
1  30.84513 -85.22907  …  4193.190458       17612.998120     3     4       3
2  43.54321 -96.63772  …  2434.234222       17505.192460     2     4       4
3  43.89744 -93.51479  …  2127.830423       12993.437350     3     5       5
4  37.59894 -76.88958  …  2113.073274        3716.525786     2     1       3

   Item4  Item5 Item6 Item7 Item8
0      2      4     3     3     4
1      4      4     4     3     3
2      4      3     4     3     3
3      3      4     5     5     5
4      3      5     3     4     3

[5 rows x 50 columns]
```

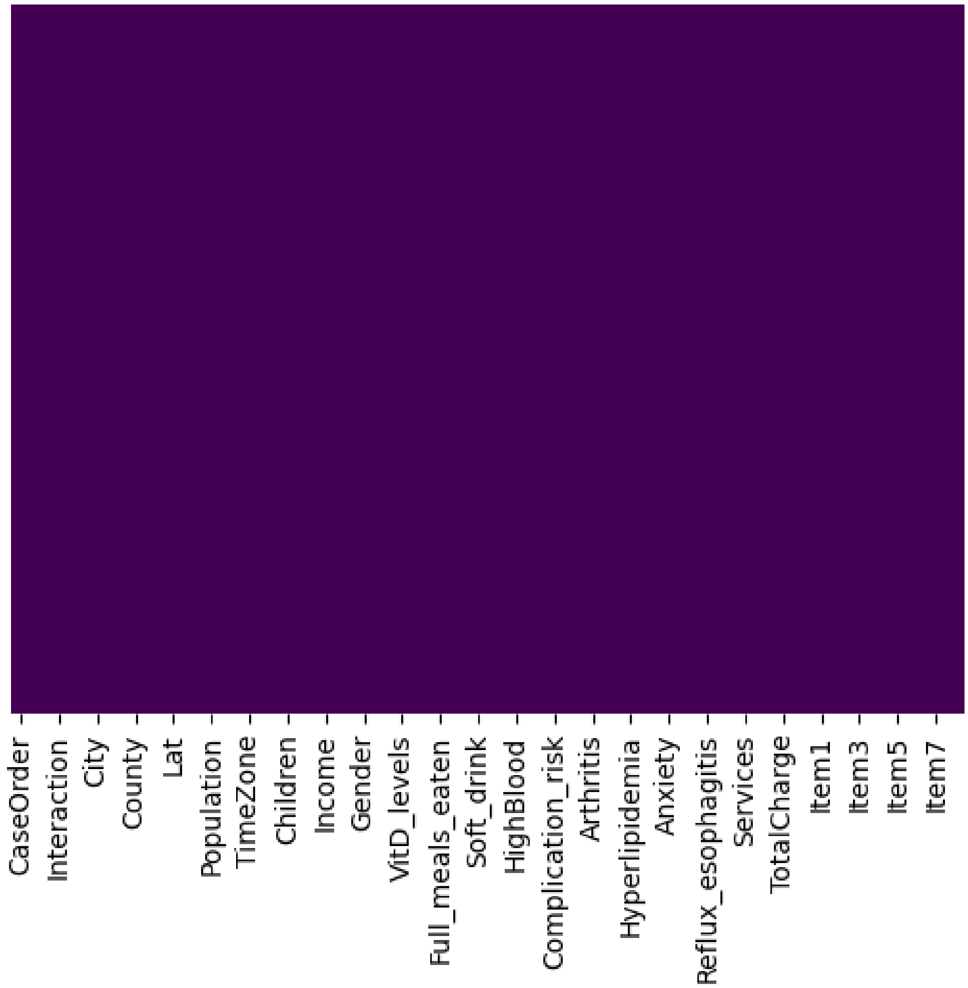
[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CaseOrder           10000 non-null  int64
 1   Customer_id         10000 non-null  object
 2   Interaction         10000 non-null  object
 3   UID                 10000 non-null  object
 4   City                10000 non-null  object
 5   State               10000 non-null  object
 6   County              10000 non-null  object
 7   Zip                 10000 non-null  int64
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64
 11  Area                10000 non-null  object
 12  TimeZone            10000 non-null  object
 13  Job                 10000 non-null  object
 14  Children            10000 non-null  int64
 15  Age                 10000 non-null  int64
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object
 18  Gender              10000 non-null  object
 19  ReAdmis             10000 non-null  object
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64
 22  Full_meals_eaten    10000 non-null  int64
 23  vitD_supp           10000 non-null  int64
 24  Soft_drink          10000 non-null  object
```

```
 25  Initial_admin       10000 non-null  object
 26  HighBlood           10000 non-null  object
 27  Stroke              10000 non-null  object
 28  Complication_risk   10000 non-null  object
 29  Overweight          10000 non-null  object
 30  Arthritis           10000 non-null  object
 31  Diabetes            10000 non-null  object
 32  Hyperlipidemia      10000 non-null  object
 33  BackPain            10000 non-null  object
 34  Anxiety             10000 non-null  object
 35  Allergic_rhinitis   10000 non-null  object
 36  Reflux_esophagitis  10000 non-null  object
 37  Asthma              10000 non-null  object
 38  Services            10000 non-null  object
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Item1               10000 non-null  int64
 43  Item2               10000 non-null  int64
 44  Item3               10000 non-null  int64
 45  Item4               10000 non-null  int64
 46  Item5               10000 non-null  int64
 47  Item6               10000 non-null  int64
 48  Item7               10000 non-null  int64
 49  Item8               10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

Look for Missing Values

[4]:
```python
# Mapping to view missing data...none present.
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis');
```



[5]:
```
df.describe()
```

[5]:
```
         CaseOrder           Zip           Lat           Lng     Population  \
count  10000.00000  10000.000000  10000.000000  10000.000000   10000.000000
mean    5000.50000  50159.323900     38.751099    -91.243080    9965.253800
std     2886.89568  27469.588208      5.403085     15.205998   14824.758614
min        1.00000    610.000000     17.967190   -174.209700       0.000000
25%     2500.75000  27592.000000     35.255120    -97.352982     694.750000
50%     5000.50000  50207.000000     39.419355    -88.397230    2769.000000
75%     7500.25000  72411.750000     42.044175    -80.438050   13945.000000
max    10000.00000  99929.000000     70.560990    -65.290170  122814.000000

           Children           Age        Income    VitD_levels    Doc_visits  \
count  10000.000000  10000.000000  10000.000000   10000.000000  10000.000000
mean       2.097200     53.511700  40490.495160      17.964262      5.012200
std        2.163659     20.638538  28521.153293       2.017231      1.045734
```

```
min        0.000000    18.000000     154.080000     9.806483     1.000000
25%        0.000000    36.000000   19598.775000    16.626439     4.000000
50%        1.000000    53.000000   33768.420000    17.951122     5.000000
75%        3.000000    71.000000   54296.402500    19.347963     6.000000
max       10.000000    89.000000  207249.100000    26.394449     9.000000

       …  TotalCharge  Additional_charges         Item1         Item2  \
count  …  10000.000000        10000.000000  10000.000000  10000.000000
mean   …   5312.172769        12934.528587      3.518800      3.506700
std    …   2180.393838         6542.601544      1.031966      1.034825
min    …   1938.312067         3125.703000      1.000000      1.000000
25%    …   3179.374015         7986.487755      3.000000      3.000000
50%    …   5213.952000        11573.977735      4.000000      3.000000
75%    …   7459.699750        15626.490000      4.000000      4.000000
max    …   9180.728000        30566.070000      8.000000      7.000000

              Item3         Item4         Item5         Item6         Item7  \
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.000000
mean       3.511100      3.515100      3.496900      3.522500      3.494000
std        1.032755      1.036282      1.030192      1.032376      1.021405
min        1.000000      1.000000      1.000000      1.000000      1.000000
25%        3.000000      3.000000      3.000000      3.000000      3.000000
50%        4.000000      4.000000      3.000000      4.000000      3.000000
75%        4.000000      4.000000      4.000000      4.000000      4.000000
max        8.000000      7.000000      7.000000      7.000000      7.000000

              Item8
count  10000.000000
mean       3.509700
std        1.042312
min        1.000000
25%        3.000000
50%        3.000000
75%        4.000000
max        7.000000

[8 rows x 23 columns]
```

0.1.3 Describe and Explore Numeric Fields:

[6]:
```python
df.describe(include = [np.number])
```

[6]:
```
         CaseOrder           Zip           Lat           Lng    Population  \
count  10000.00000  10000.000000  10000.000000  10000.000000  10000.000000
mean    5000.50000  50159.323900     38.751099    -91.243080   9965.253800
std     2886.89568  27469.588208      5.403085     15.205998  14824.758614
min        1.00000    610.000000     17.967190   -174.209700      0.000000
```

```
25%     2500.75000  27592.000000     35.255120    -97.352982     694.750000
50%     5000.50000  50207.000000     39.419355    -88.397230    2769.000000
75%     7500.25000  72411.750000     42.044175    -80.438050   13945.000000
max    10000.00000  99929.000000     70.560990    -65.290170  122814.000000

           Children           Age         Income   VitD_levels    Doc_visits  \
count  10000.000000  10000.000000   10000.000000  10000.000000  10000.000000
mean       2.097200     53.511700   40490.495160     17.964262      5.012200
std        2.163659     20.638538   28521.153293      2.017231      1.045734
min        0.000000     18.000000     154.080000      9.806483      1.000000
25%        0.000000     36.000000   19598.775000     16.626439      4.000000
50%        1.000000     53.000000   33768.420000     17.951122      5.000000
75%        3.000000     71.000000   54296.402500     19.347963      6.000000
max       10.000000     89.000000  207249.100000     26.394449      9.000000

       …   TotalCharge  Additional_charges         Item1         Item2  \
count  …  10000.000000        10000.000000  10000.000000  10000.000000
mean   …   5312.172769        12934.528587      3.518800      3.506700
std    …   2180.393838         6542.601544      1.031966      1.034825
min    …   1938.312067         3125.703000      1.000000      1.000000
25%    …   3179.374015         7986.487755      3.000000      3.000000
50%    …   5213.952000        11573.977735      4.000000      3.000000
75%    …   7459.699750        15626.490000      4.000000      4.000000
max    …   9180.728000        30566.070000      8.000000      7.000000

              Item3         Item4         Item5         Item6         Item7  \
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.000000
mean       3.511100      3.515100      3.496900      3.522500      3.494000
std        1.032755      1.036282      1.030192      1.032376      1.021405
min        1.000000      1.000000      1.000000      1.000000      1.000000
25%        3.000000      3.000000      3.000000      3.000000      3.000000
50%        4.000000      4.000000      3.000000      4.000000      3.000000
75%        4.000000      4.000000      4.000000      4.000000      4.000000
max        8.000000      7.000000      7.000000      7.000000      7.000000

              Item8
count  10000.000000
mean       3.509700
std        1.042312
min        1.000000
25%        3.000000
50%        3.000000
75%        4.000000
max        7.000000

[8 rows x 23 columns]
```

[7]:
```
df_num = df.select_dtypes(include='number')
df_num.head()
```

[7]:
```
   CaseOrder    Zip       Lat       Lng  Population  Children  Age    Income  \
0          1  35621  34.34960 -86.72508        2951         1   53  86575.93
1          2  32446  30.84513 -85.22907       11303         3   51  46805.99
2          3  57110  43.54321 -96.63772       17125         3   53  14370.14
3          4  56072  43.89744 -93.51479        2162         0   78  39741.49
4          5  23181  37.59894 -76.88958        5287         1   22   1209.56

   VitD_levels  Doc_visits  …  TotalCharge  Additional_charges  Item1  \
0    19.141466           6  …  3726.702860        17939.403420      3
1    18.940352           4  …  4193.190458        17612.998120      3
2    18.057507           4  …  2434.234222        17505.192460      2
3    16.576858           4  …  2127.830423        12993.437350      3
4    17.439069           5  …  2113.073274         3716.525786      2

   Item2  Item3  Item4  Item5  Item6  Item7  Item8
0      3      2      2      4      3      3      4
1      4      3      4      4      4      3      3
2      4      4      4      3      4      3      3
3      5      5      3      4      5      5      5
4      1      3      3      5      3      4      3

[5 rows x 23 columns]
```

0.1.4 Describe and Explore Categorical Fields:

[8]:
```
df.describe(exclude = [np.number])
```

[8]:
```
       Customer_id                           Interaction  \
count        10000                                 10000
unique       10000                                 10000
top        C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
freq             1                                     1

                                     UID     City  State     County   Area  \
count                              10000    10000  10000      10000  10000
unique                             10000     6072     52       1607      3
top     3a83ddb66e2ae73798bdf1d705dc0932  Houston     TX  Jefferson  Rural
freq                                   1       36    553        118   3369

                TimeZone                                 Job  Marital  …  \
count              10000                               10000    10000  …
unique                26                                 639        5  …
top     America/New_York  Outdoor activities/education manager  Widowed  …
freq                3889                                  29     2045  …
```

```
       Overweight Arthritis Diabetes Hyperlipidemia BackPain Anxiety  \
count       10000      10000    10000          10000    10000   10000
unique          2          2        2              2        2       2
top           Yes         No       No             No       No      No
freq         7094       6426     7262           6628     5886    6785

       Allergic_rhinitis Reflux_esophagitis Asthma    Services
count              10000              10000  10000       10000
unique                 2                  2      2           4
top                   No                 No     No  Blood Work
freq                6059               5865   7107        5265

[4 rows x 27 columns]
```

[9]:
```python
df_cat = df.select_dtypes(exclude='number')
df_cat.head()
```

[9]:
```
  Customer_id                           Interaction  \
0     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
1     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
2     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
3     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
4     C544523  5885f56b-d6da-43a3-8760-83583af94266

                                UID          City State         County  \
0  3a83ddb66e2ae73798bdf1d705dc0932           Eva    AL         Morgan
1  176354c5eef714957d486009feabf195      Marianna    FL        Jackson
2  e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD      Minnehaha
3  cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN         Waseca
4  d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William

       Area          TimeZone                              Job   Marital  \
0  Suburban   America/Chicago  Psychologist, sport and exercise  Divorced
1     Urban   America/Chicago      Community development worker   Married
2  Suburban   America/Chicago           Chief Executive Officer   Widowed
3  Suburban   America/Chicago               Early years teacher   Married
4     Rural  America/New_York       Health promotion specialist   Widowed

   … Overweight Arthritis Diabetes Hyperlipidemia BackPain Anxiety  \
0  …         No        Yes      Yes             No      Yes     Yes
1  …        Yes         No       No             No       No      No
2  …        Yes         No      Yes             No       No      No
3  …         No        Yes       No             No       No      No
4  …         No         No       No            Yes       No      No

  Allergic_rhinitis Reflux_esophagitis Asthma    Services
```

```
0              Yes                 No    Yes    Blood Work
1               No                Yes     No   Intravenous
2               No                 No     No    Blood Work
3               No                Yes    Yes    Blood Work
4              Yes                 No     No       CT Scan

[5 rows x 27 columns]
```

[10]:
```python
df[['ReAdmis']].describe()
df
```

[10]:
```
      CaseOrder Customer_id                           Interaction  \
0             1     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
1             2     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
2             3     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
3             4     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
4             5     C544523  5885f56b-d6da-43a3-8760-83583af94266
…           …         …                                     …
9995       9996     B863060  a25b594d-0328-486f-a9b9-0567eb0f9723
9996       9997     P712040  70711574-f7b1-4a17-b15f-48c54564b70f
9997       9998     R778890  1d79569d-8e0f-4180-a207-d67ee4527d26
9998       9999     E344109  f5a68e69-2a60-409b-a92f-ac0847b27db0
9999      10000     I569847  bc482c02-f8c9-4423-99de-3db5e62a18d5

                                   UID          City State        County  \
0     3a83ddb66e2ae73798bdf1d705dc0932           Eva    AL        Morgan
1     176354c5eef714957d486009feabf195      Marianna    FL       Jackson
2     e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD     Minnehaha
3     cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca
4     d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William
…                                  …           …   …           …
9995  39184dc28cc038871912ccc4500049e5       Norlina    NC        Warren
9996  3cd124ccd43147404292e883bf9ec55c       Milmay    NJ      Atlantic
9997  41b770aeee97a5b9e7f69c906a8119d7     Southside    TN    Montgomery
9998  2bb491ef5b1beb1fed758cc6885c167a        Quinn    SD    Pennington
9999  95663a202338000abdf7e09311c2a8a1    Coraopolis    PA     Allegheny

        Zip       Lat       Lng  …  TotalCharge Additional_charges Item1  \
0     35621  34.34960 -86.72508  …  3726.702860       17939.403420     3
1     32446  30.84513 -85.22907  …  4193.190458       17612.998120     3
2     57110  43.54321 -96.63772  …  2434.234222       17505.192460     2
3     56072  43.89744 -93.51479  …  2127.830423       12993.437350     3
4     23181  37.59894 -76.88958  …  2113.073274        3716.525786     2
…       …       …       …  …  …               …                …   …
9995  27563  36.42886 -78.23716  …  6850.942000        8927.642000     3
9996   8340  39.43609 -74.87302  …  7741.690000       28507.150000     3
9997  37171  36.36655 -87.29988  …  8276.481000       15281.210000     3
```

```
9998  57775  44.10354 -102.01590  …  7644.483000        7781.678000     5
9999  15108  40.49998  -80.19959  …  7887.553000       11643.190000     4

     Item2  Item3  Item4  Item5 Item6 Item7 Item8
0        3      2      2      4     3     3     4
1        4      3      4      4     4     3     3
2        4      4      4      3     4     3     3
3        5      5      3      4     5     5     5
4        1      3      3      5     3     4     3
…    …    …    …    …    …    …    …
9995     2      2      3      4     3     4     2
9996     3      4      2      5     3     4     4
9997     3      3      4      4     2     3     2
9998     5      3      4      4     3     4     3
9999     3      3      2      3     6     4     3

[10000 rows x 50 columns]
```

[11]:
```python
df['ReAdmis_Yes'] = df['ReAdmis']
df['ReAdmis_Yes'] = df['ReAdmis_Yes'].eq('Yes').astype(int)
df['ReAdmis_Yes']
```

[11]:
```
0       0
1       0
2       0
3       0
4       0
       ..
9995    0
9996    1
9997    1
9998    1
9999    1
Name: ReAdmis_Yes, Length: 10000, dtype: int64
```

[12]:
```python
df.columns
```

[12]:
```
Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
       'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',
       'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Services', 'Initial_days',
       'TotalCharge', 'Additional_charges', 'Item1', 'Item2', 'Item3', 'Item4',
       'Item5', 'Item6', 'Item7', 'Item8', 'ReAdmis_Yes'],
```

```
      dtype='object')
```

[13]:
```python
df_num.columns
```

[13]:
```
Index(['CaseOrder', 'Zip', 'Lat', 'Lng', 'Population', 'Children', 'Age',
       'Income', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Initial_days', 'TotalCharge', 'Additional_charges', 'Item1', 'Item2',
       'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

[14]:
```python
df['Gender'].value_counts()
```

[14]:
```
Female       5018
Male         4768
Nonbinary     214
Name: Gender, dtype: int64
```

[15]:
```python
df['Initial_admin'].value_counts()
```

[15]:
```
Emergency Admission      5060
Elective Admission       2504
Observation Admission    2436
Name: Initial_admin, dtype: int64
```

0.2 Part 1: Research Question:

0.2.1 [A1] Question: "From information about previous patients who were readmitted, can we predict which patients are likely to be readmitted in the future?"

0.2.2 Prune Data

[16]:
```python
df.columns
```

[16]:
```
Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
       'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',
       'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Services', 'Initial_days',
       'TotalCharge', 'Additional_charges', 'Item1', 'Item2', 'Item3', 'Item4',
       'Item5', 'Item6', 'Item7', 'Item8', 'ReAdmis_Yes'],
      dtype='object')
```

[17]:
```
# Start pruning non-relavent series
pruned_df = df.drop(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City',␣
↪'State', \
                    'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area',␣
↪'TimeZone', 'Job', \
                    'Children', 'Income', 'Marital', 'VitD_levels',␣
↪'Full_meals_eaten', \
                    'Soft_drink', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5',␣
↪'Item6', 'Item7', 'Item8'], axis=1)
pruned_df.head()
```

[17]:
```
   Age  Gender ReAdmis  Doc_visits  vitD_supp        Initial_admin HighBlood  \
0   53    Male      No           6          0  Emergency Admission       Yes
1   51  Female      No           4          1  Emergency Admission       Yes
2   53  Female      No           4          0   Elective Admission       Yes
3   78    Male      No           4          0   Elective Admission        No
4   22  Female      No           5          2   Elective Admission        No

  Stroke Complication_risk Overweight  … BackPain Anxiety Allergic_rhinitis  \
0     No            Medium         No  …      Yes     Yes               Yes
1     No              High        Yes  …       No      No                No
2     No            Medium        Yes  …       No      No                No
3    Yes            Medium         No  …       No      No                No
4     No               Low         No  …       No      No               Yes

  Reflux_esophagitis Asthma     Services Initial_days  TotalCharge  \
0                 No    Yes   Blood Work    10.585770  3726.702860
1                Yes     No  Intravenous    15.129562  4193.190458
2                 No     No   Blood Work     4.772177  2434.234222
3                Yes    Yes   Blood Work     1.714879  2127.830423
4                 No     No      CT Scan     1.254807  2113.073274

  Additional_charges  ReAdmis_Yes
0       17939.403420            0
1       17612.998120            0
2       17505.192460            0
3       12993.437350            0
4        3716.525786            0

[5 rows x 23 columns]
```

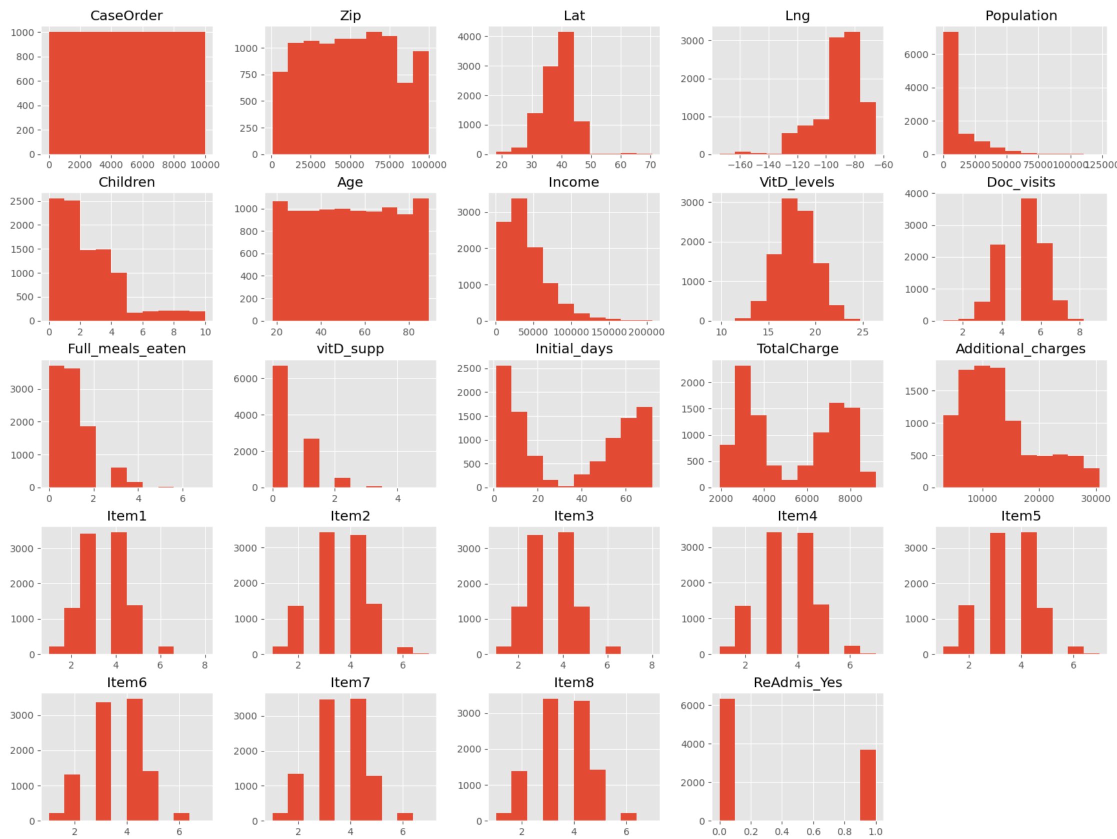
[18]:
```
pruned_df.columns
```

[18]:
```
Index(['Age', 'Gender', 'ReAdmis', 'Doc_visits', 'vitD_supp', 'Initial_admin',
       'HighBlood', 'Stroke', 'Complication_risk', 'Overweight', 'Arthritis',
       'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',
       'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Services',
```

```
       'Initial_days', 'TotalCharge', 'Additional_charges', 'ReAdmis_Yes'],
      dtype='object')
```

[19]:
```
# https://www.datacamp.com/community/tutorials/
 ↪preprocessing-in-data-science-part-1-centering-scaling-and-knn
plt.style.use('ggplot')
# df = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/
 ↪wine-quality/winequality-red.csv ' , sep = ';')
X = pruned_df.drop('ReAdmis_Yes', 1).values # drop target variable
y1 = pruned_df['ReAdmis_Yes'].values
pd.DataFrame.hist(df, figsize = [20,15]);
```

```
/var/folders/45/_087y05165x0c7wb_dw4k6nh0000gn/T/ipykernel_60369/2260330480.py:4
: FutureWarning: In a future version of pandas all arguments of DataFrame.drop
except for the argument 'labels' will be keyword-only.
  X = pruned_df.drop('ReAdmis_Yes', 1).values # drop target variable
```



[20]:
```
# https://realpython.com/knn-python/
# Correlations with target?
correlation_matrix = pruned_df.corr()
```

```
print(correlation_matrix["ReAdmis_Yes"] > 0.5)
```

```
Age                   False
Doc_visits            False
vitD_supp             False
Initial_days           True
TotalCharge            True
Additional_charges    False
ReAdmis_Yes            True
Name: ReAdmis_Yes, dtype: bool
```

```
/var/folders/45/_087y05165x0c7wb_dw4k6nh0000gn/T/ipykernel_60369/2874329755.py:3
: FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = pruned_df.corr()
```

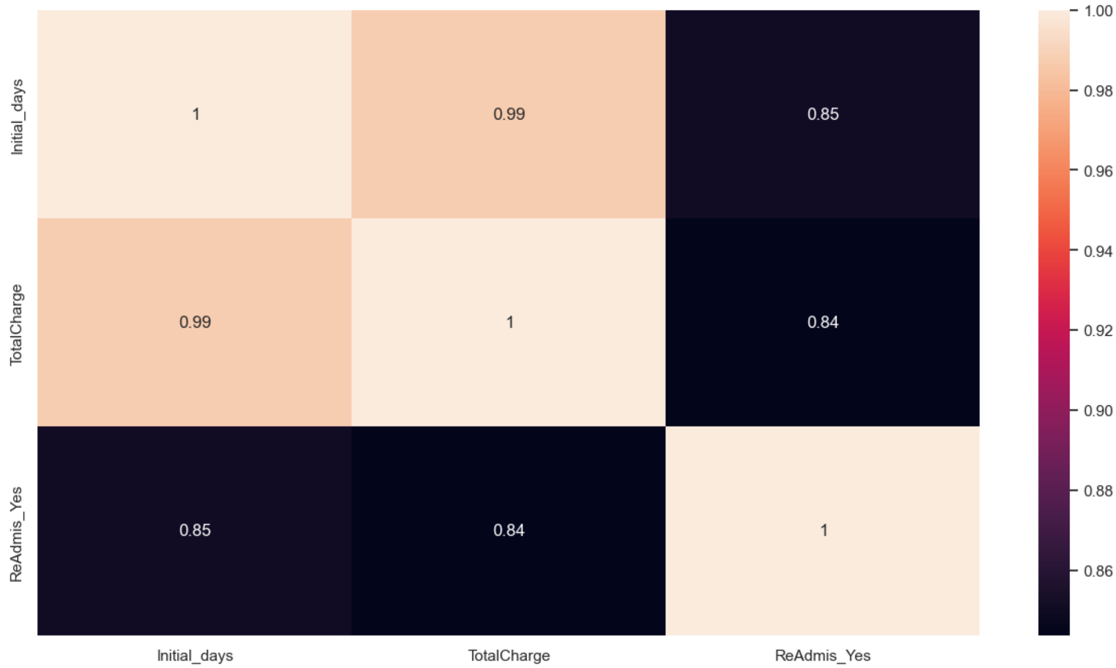
[D1] Construct an initial multiple regression model from *all* predictors that were identified in [C2]

[21]:
```
pruned_df = pruned_df[['Initial_days', 'TotalCharge', 'ReAdmis_Yes']]
```

[22]:
```
pruned_df.shape
```

[22]:
```
(10000, 3)
```

[23]:
```
# Trying to make sense of numerical values, discover possible correlations
# Ref1: https://www.geeksforgeeks.org/
 ↪how-to-create-a-seaborn-correlation-heatmap-in-python/
# Ref2: https://medium.com/@szabo.bibor/
 ↪how-to-create-a-seaborn-correlation-heatmap-in-python-834c0686b88e
sns.set(rc = {'figure.figsize':(15,8)})
sns.heatmap(pruned_df.corr(), annot=True);
```



0.3 KNN

0.3.1 Define Distances on the Vectors of Independent Vars

[24]:
```
# Independed Var
# pruned_df = pruned_df[['Initial_days', 'TotalCharge', 'ReAdmis_Yes']]
X = pruned_df.drop('ReAdmis_Yes', axis=1)
X = X.values

# Dependent Var
y = pruned_df['ReAdmis_Yes']
y = y.values
```

0.4 Confusion Matrix

0.4.1 Supervised Learning: To Predict Target Variable (ReAdmi_Yes) Given Predictor Vars

Classification: Target Variable is Catagorical

Features –> Predictor Vars –> Independent Vars

Target Vars –> Dependent Vars –> Response Vars

[25]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
from sklearn.metrics import confusion_matrix

knn = KNeighborsClassifier(n_neighbors=3)  # From Fit kNN Regression Below

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3,
    random_state=73,
    stratify=y)  # stratisfy reflects labels of data for both train and test
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred));
```

```
Test set predictions:
 [0 1 0 … 0 1 0]
```

[26]:
```
#accuracy of Model
knn.score(X_test, y_test)
```

[26]: 0.9766666666666667

[27]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1899
           1       0.97      0.97      0.97      1101

    accuracy                           0.98      3000
   macro avg       0.98      0.97      0.97      3000
weighted avg       0.98      0.98      0.98      3000
```

0.5 Preprocessing - Scaling

[28]:
```
from sklearn.preprocessing import scale
Xs = scale(X)
Xs_train, Xs_test, y_train, y_test = train_test_split(Xs, y, test_size=0.3,␣
 ↪random_state=73)
knn_model_2 = knn.fit(Xs_train, y_train)
print('k-NN score for test set: %f' % knn_model_2.score(Xs_test, y_test))
print('k-NN score for training set: %f' % knn_model_2.score(Xs_train, y_train))
y_true, y_pred = y_test, knn_model_2.predict(Xs_test)
print(classification_report(y_true, y_pred))
```

```
k-NN score for test set: 0.980333
k-NN score for training set: 0.986143
              precision    recall  f1-score   support
```

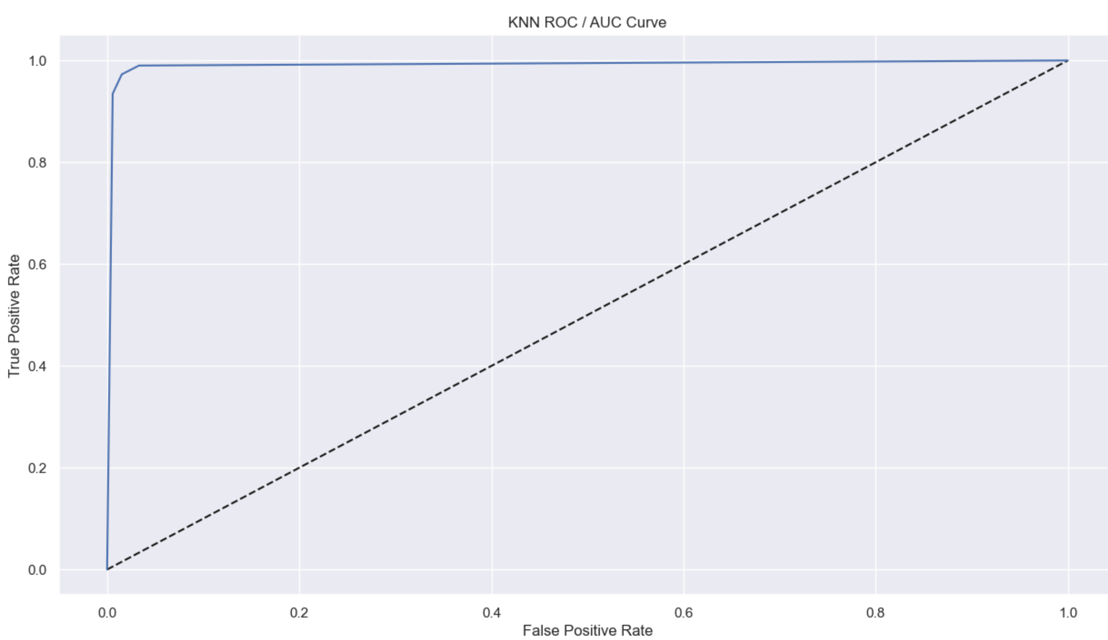
```
           0       0.98      0.98      0.98      1899
           1       0.97      0.97      0.97      1101

    accuracy                           0.98      3000
   macro avg       0.98      0.98      0.98      3000
weighted avg       0.98      0.98      0.98      3000
```

[29]:
```
print('k-NN score for test set: %f' % knn_model_2.score(Xs_test, y_test))
print('k-NN score for training set: %f' % knn_model_2.score(Xs_train, y_train))
y_true, y_pred = y_test, knn_model_2.predict(Xs_test)
print(confusion_matrix(y_test, y_pred))
```

```
k-NN score for test set: 0.980333
k-NN score for training set: 0.986143
[[1870   29]
 [  30 1071]]
```

0.6 Logistic Regression, Probability Thresholds and ROC Curve

[30]:
```
from sklearn.metrics import roc_curve
y_pred_prob = knn.predict_proba(Xs_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='KNN Regression')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('KNN ROC / AUC Curve')
plt.show();
```



[31]:
```
knn.predict_proba(Xs_test)[:,1]
```

[31]: array([0., 1., 0., …, 0., 0., 0.])

[32]:
```
from sklearn.metrics import roc_auc_score

y_pred_prob = knn.predict_proba(Xs_test)[:,1]
roc_auc_score(y_test, y_pred_prob)
```

[32]: 0.9913121730018046

[33]:
```
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(knn, X, y, cv=5,
                                scoring='roc_auc')
print(cv_scores)
```

```
[0.99454297 0.99455041 0.99318801 0.99310461 0.82710784]
```

[34]:
```
# Optimal KNN
from sklearn.model_selection import GridSearchCV
param_grid = {'n_neighbors': np.arange(1, 100)}
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, param_grid, cv=5)
knn_cv.fit(X, y)
knn_cv.best_params_
```

[34]: {'n_neighbors': 2}

[35]:
```
knn_cv.best_score_
```

[35]: 0.9273999999999999

0.7 Prediction for Classification

[36]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=73)
rfc.fit(Xs_train, y_train)
rfc.predict(Xs_test)
```

[36]: array([0, 1, 0, …, 0, 0, 0])

[37]:
```
pd.Series(rfc.predict(Xs_test)).value_counts()
```

[37]:
```
0    1907
1    1093
```

```
dtype: int64
```

[38]:
```
rfc.predict_proba(Xs_test)
```

[38]:
```
array([[1., 0.],
       [0., 1.],
       [1., 0.],
       …,
       [1., 0.],
       [1., 0.],
       [1., 0.]])
```

[39]:
```
rfc = RandomForestClassifier(random_state=73)
rfc.get_params()
```

[39]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 73,
 'verbose': 0,
 'warm_start': False}
```

[40]:
```
rfc.fit(Xs_train, y_train)
rfc.score(Xs_test, y_test)
```

[40]:
```
0.98
```

[41]:
```
# Using scikit-learn to Inspect Model Fit
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import mean_absolute_error

train_preds = knn_model_2.predict(Xs_train)
mse = mean_squared_error(y_train, train_preds)
print("MSE: ", mse)
```

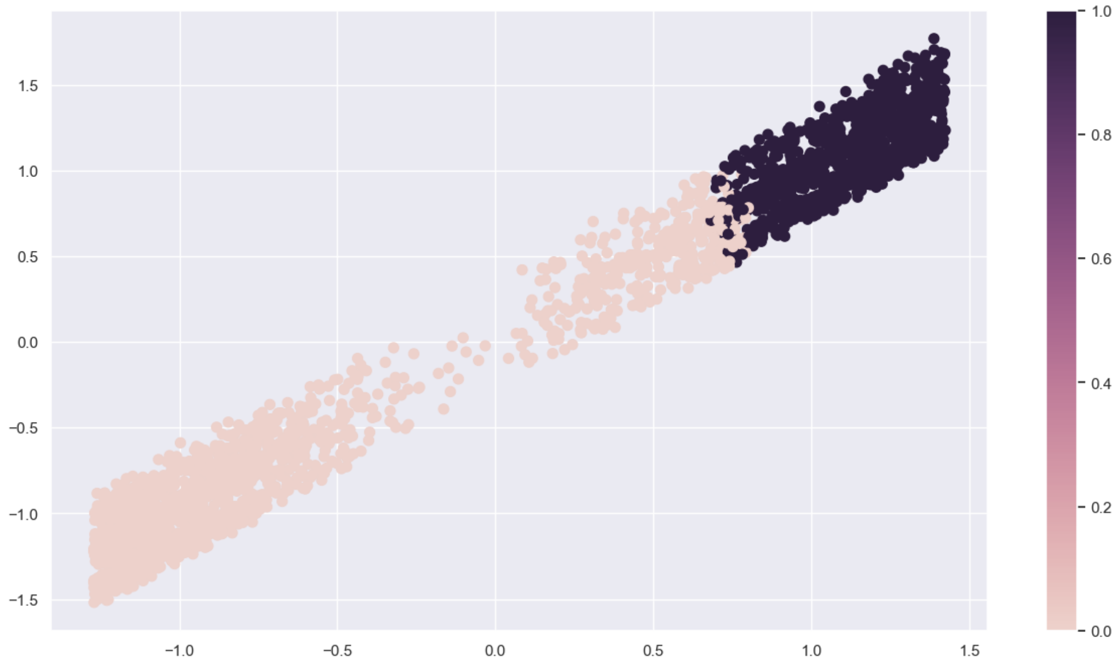
```
test_preds = knn_model_2.predict(Xs_test)
mae = mean_absolute_error(y_test, test_preds)
print("MAE: ", mae)
```

```
MSE:  0.013857142857142858
MAE:  0.019666666666666666
```

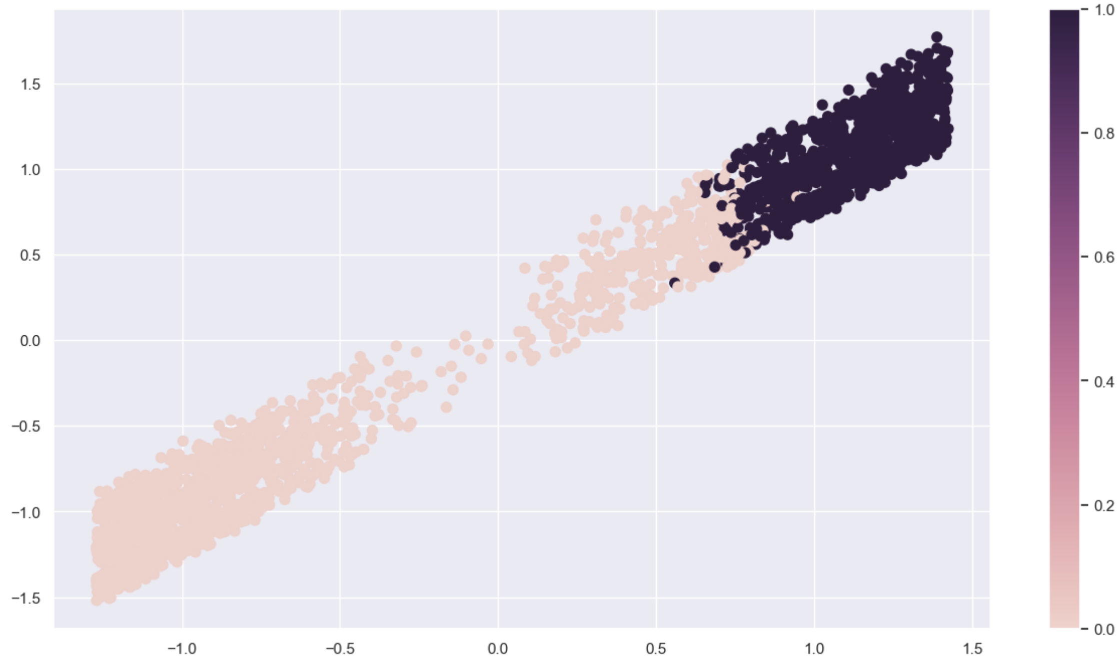
[42]:
```
test_preds = knn_model_2.predict(Xs_test)
mse = mean_squared_error(y_test, test_preds)
mse
#rmse = sqrt(mse)
#rmse
```

[42]: 0.019666666666666666

[43]:
```
# Predicted
import seaborn as sns
cmap = sns.cubehelix_palette(as_cmap=True)
f, ax = plt.subplots()
points = ax.scatter(
    Xs_test[:, 0], Xs_test[:, 1], c=test_preds, s=50, cmap=cmap
)
f.colorbar(points)
plt.show()
```



[44]:
```python
# Actual
import seaborn as sns
cmap = sns.cubehelix_palette(as_cmap=True)
f, ax = plt.subplots()
points = ax.scatter(
    Xs_test[:, 0], Xs_test[:, 1], c=y_test, s=50, cmap=cmap
)
f.colorbar(points)
plt.show()
```



[45]:
```python
# Mean Absolute Error (MAE)
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, y_pred)
```

[45]: 0.019666666666666666

[46]:
```python
# Mean Squared Error (MSE)
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
```

[46]: 0.019666666666666666

[47]:
```python
 from sklearn.metrics import accuracy_score, precision_score, recall_score
accuracy_score(y_test, y_pred)
```

[47]: 0.9803333333333333

[48]:
```
precision_score(y_test, y_pred)
```

[48]: 0.9736363636363636

[49]:
```
recall_score(y_test, y_pred)
```

[49]: 0.9727520435967303

[50]:
```
pruned_df
```

[50]:

	Initial_days	TotalCharge	ReAdmis_Yes
0	10.585770	3726.702860	0
1	15.129562	4193.190458	0
2	4.772177	2434.234222	0
3	1.714879	2127.830423	0
4	1.254807	2113.073274	0
…	…	…	…
9995	51.561220	6850.942000	0
9996	68.668240	7741.690000	1
9997	70.154180	8276.481000	1
9998	63.356900	7644.483000	1
9999	70.850590	7887.553000	1

[10000 rows x 3 columns]

0.7.1 Export Data

[51]:
```
[['Initial_days', 'TotalCharge', 'ReAdmis_Yes']]
```

[51]: [['Initial_days', 'TotalCharge', 'ReAdmis_Yes']]

[52]:
```
X_train.shape
```

[52]: (7000, 2)

[53]:
```
# Export Training Data
X_train_data = pd.DataFrame(X_train, columns = ['Initial_days','TotalCharge'])
y_train_data = pd.DataFrame(y_train, columns = ['ReAdmis_Yes'])
```

[54]:
```
# Export Testing Data
X_test_data = pd.DataFrame(X_test, columns = ['Initial_days','TotalCharge'])
y_test_data = pd.DataFrame(y_test, columns = ['ReAdmis_Yes'])
```

[55]:
```
pruned_df.to_csv('final_cleaned_dataset.csv', index=False)
X_train_data.to_csv('X_train_data.csv', index=False)
```

```
X_test_data.to_csv('X_test_data.csv', index=False)
y_train_data.to_csv('y_train_data.csv', index=False)
y_test_data.to_csv('y_test_data.csv', index=False)
```

[56]:
```
pruned_df.describe()
```

[56]:

	Initial_days	TotalCharge	ReAdmis_Yes
count	10000.000000	10000.000000	10000.000000
mean	34.455299	5312.172769	0.366900
std	26.309341	2180.393838	0.481983
min	1.001981	1938.312067	0.000000
25%	7.896215	3179.374015	0.000000
50%	35.836244	5213.952000	0.000000
75%	61.161020	7459.699750	1.000000
max	71.981490	9180.728000	1.000000

[57]:
```
pruned_df.nunique()
```

[57]:
```
Initial_days    9997
TotalCharge     9997
ReAdmis_Yes        2
dtype: int64
```

[58]:
```
pruned_df['ReAdmis_Yes'].value_counts()
```

[58]:
```
0    6331
1    3669
Name: ReAdmis_Yes, dtype: int64
```