JWillis_D208_PA1_MultipleRegression

January 8, 2023

0.1 D208 - Predictive Modeling - PA1

0.1.1 Import Libraries

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as plt
%matplotlib inline
import statsmodels.api as sm
from pandas import DataFrame
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]
```

Start understanding data

```
[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
```

[4]:
```
df.describe()
```

[4]:
```
          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]
```

[5]:
```
df['Gender'].value_counts()
```

[5]:
```
Female       5018
Male         4768
Nonbinary     214
Name: Gender, dtype: int64
```

[6]:
```
df.columns
```

[6]:
```
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'],
      dtype='object')
```

0.1.3 Any Rows With Nulls?

[7]:
```
print("Are there any rows with nulls: " + str(df.isnull().all(axis=1).any()))
```

```
Are there any rows with nulls: False
```

0.1.4 Any Missing Values?

[8]:
```
df.loc[:, df.isnull().any()]
```

[8]:
```
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99,
…]

[10000 rows x 0 columns]
```

0.2 Part 1: Research Question:

0.2.1 [A1] Question: "Can the following three features (Initial Days, Readmission, and Diabetes) help predict total charges?"

Describe & Explore Numeric Fields:

[9]:
```
#https://stackoverflow.com/questions/24524104/
↪pandas-describe-is-not-returning-summary-of-all-columns
# Describe Numeric Fields
df.describe(include = [np.number])
```

[9]:
```
         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]
```

[10]:
```
# df_num = df.select_dtypes(include='number')
# df_num.head()
df_num = df.select_dtypes(include='number')
df_num.head()
```

[10]:
```
   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]
```

Describe & Explore Categorical Fields:

[11]:
```
# Describe Categorical Fields
df.describe(include = ['O'])
```

[11]:
```
       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]
```

[12]:
```
df_cat = df.select_dtypes(exclude='number')
df_cat.head()
```

[12]:
```
  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]
```

[B cont.] Create Subset Data Group to Focus On and Describe

[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_cat.columns
```

[14]:
```
Index(['Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Area',
       'TimeZone', 'Job', 'Marital', 'Gender', 'ReAdmis', 'Soft_drink',
       'Initial_admin', 'HighBlood', 'Stroke', 'Complication_risk',
       'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
       'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
       'Services'],
      dtype='object')
```

Prune Numerical Fields

Add Columns to Quantify Boolean Fields

[15]:
```python
pruned_df_num = df_num.drop(['CaseOrder', 'Population', 'Children', 'Income',␣
 ↪'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp','Zip', 'Lat',␣
 ↪'Lng', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7',␣
 ↪'Item8'], axis=1)
# Transform & Add Quantified Data Fields As Needed:
# pruned_df_num['Overweight_Num'] = df['Overweight'].eq('Yes').astype(int)
pruned_df_num['Diabetes_Num'] = df['Diabetes'].eq('Yes').astype(int)
pruned_df_num['ReAdmis_Num'] = df['ReAdmis'].eq('Yes').astype(int)
# pruned_df_num['Gender_Num'] = df['Gender'].eq('Male').astype(int)

pruned_df_num
```

[15]:
```
      Age  Initial_days  TotalCharge  Additional_charges  Diabetes_Num  \
0      53     10.585770  3726.702860        17939.403420             1
1      51     15.129562  4193.190458        17612.998120             0
2      53      4.772177  2434.234222        17505.192460             1
3      78      1.714879  2127.830423        12993.437350             0
4      22      1.254807  2113.073274         3716.525786             0
…     …             …            …                 …           …
9995   25     51.561220  6850.942000         8927.642000             0
9996   87     68.668240  7741.690000        28507.150000             1
9997   45     70.154180  8276.481000        15281.210000             0
9998   43     63.356900  7644.483000         7781.678000             0
9999   70     70.850590  7887.553000        11643.190000             0

      ReAdmis_Num
0               0
1               0
2               0
3               0
4               0
…             …
9995            0
9996            1
9997            1
9998            1
9999            1

[10000 rows x 6 columns]
```

Prune Categorical Fields

[16]:
```python
pruned_df_cat = df_cat.drop(['Customer_id', 'Interaction', 'UID', 'City',␣
 ↪'State', 'County', 'Area', 'TimeZone', 'Job', \
                              'Marital', 'ReAdmis', 'Diabetes',␣
 ↪'Overweight','Soft_drink', 'BackPain', 'Anxiety', 'Allergic_rhinitis',␣
 ↪'Reflux_esophagitis', 'Asthma'], axis=1)
pruned_df_cat
```

[16]:
```
      Gender        Initial_admin HighBlood Stroke Complication_risk  \
0       Male  Emergency Admission       Yes     No            Medium
1     Female  Emergency Admission       Yes     No              High
2     Female   Elective Admission       Yes     No            Medium
3       Male   Elective Admission        No    Yes            Medium
4     Female   Elective Admission        No     No               Low
…        …                  …         …      …                 …
9995    Male  Emergency Admission       Yes     No            Medium
9996    Male   Elective Admission       Yes     No            Medium
9997  Female   Elective Admission       Yes     No              High
```

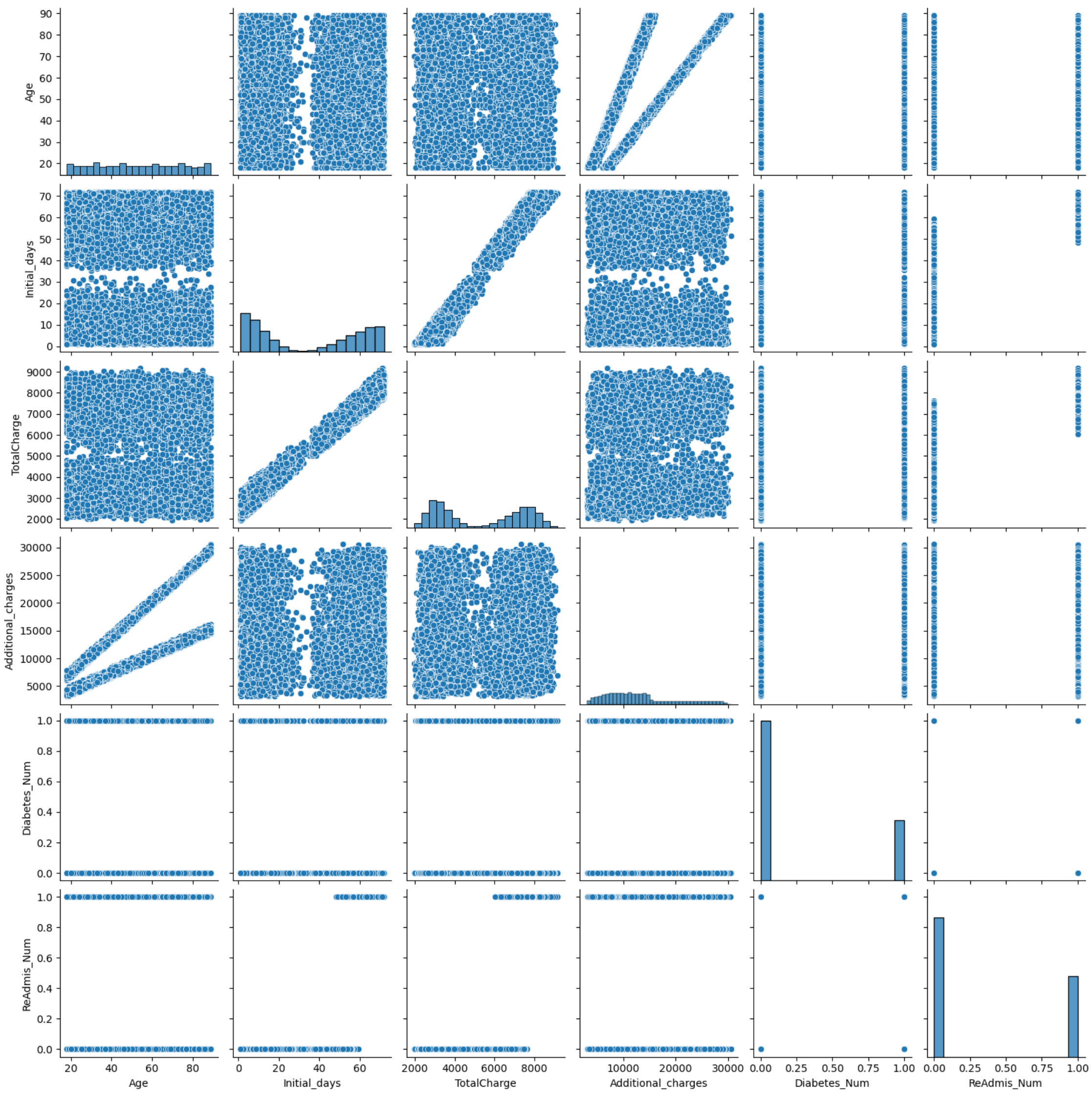
```
9998    Male    Emergency Admission        No      No             Medium
9999  Female  Observation Admission        No      No                Low

     Arthritis Hyperlipidemia     Services
0          Yes             No   Blood Work
1           No             No  Intravenous
2           No             No   Blood Work
3          Yes             No   Blood Work
4           No            Yes      CT Scan
…         …             …          …
9995        No             No  Intravenous
9996       Yes             No      CT Scan
9997        No             No  Intravenous
9998        No             No   Blood Work
9999       Yes            Yes   Blood Work

[10000 rows x 8 columns]
```

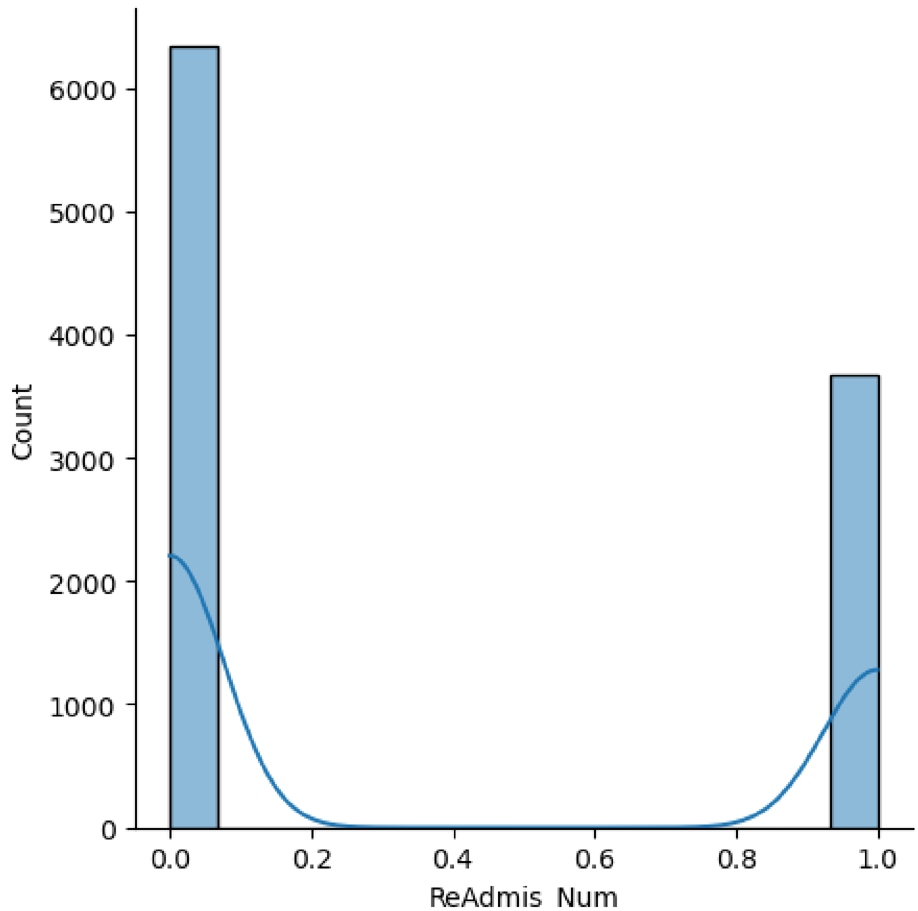
Plot Data

[17]:
```
sns.pairplot(pruned_df_num);
```

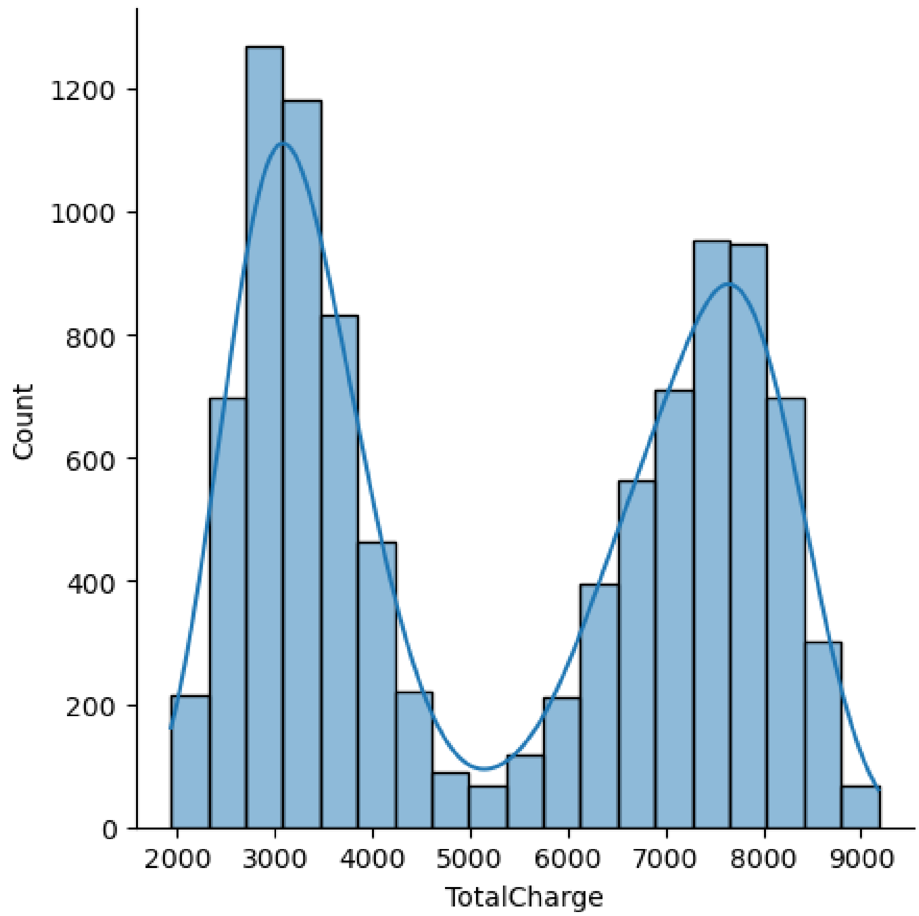


0.2.2 Univariate Analysis

[18]:
```python
# https://seaborn.pydata.org/generated/seaborn.displot.html
# Distribution of Total Charge
sns.displot(data=pruned_df_num,
            x='ReAdmis_Num',
            kde=True);
```

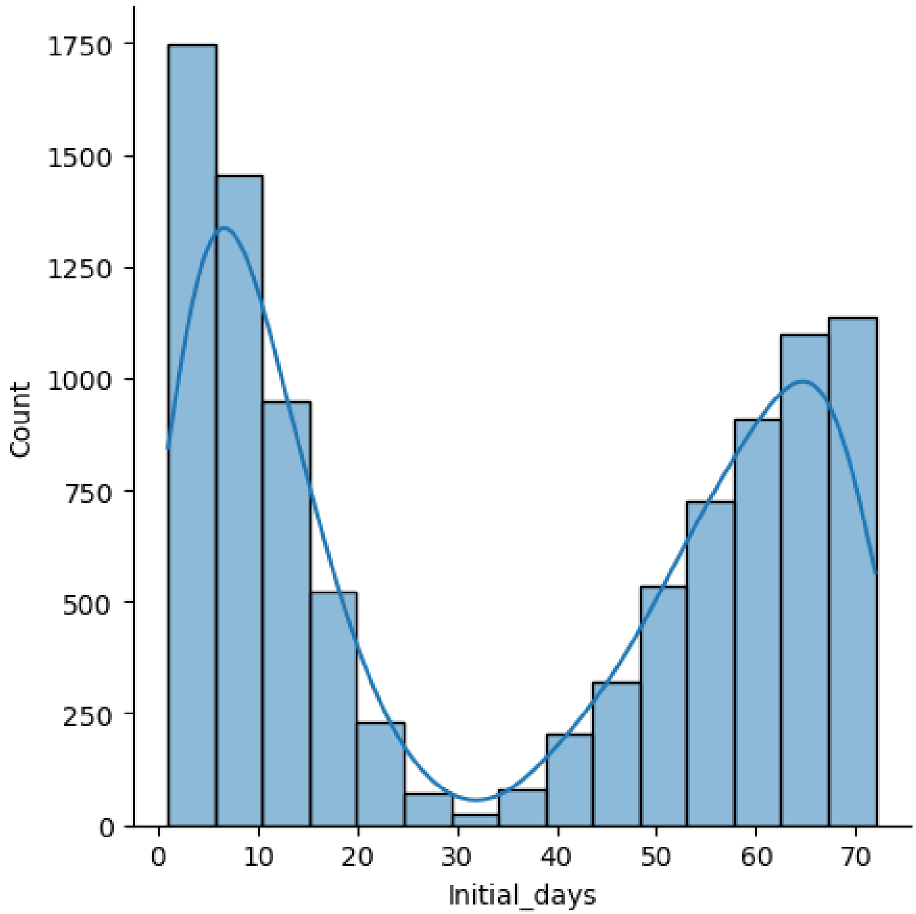


[19]:
```
sns.displot(data=pruned_df_num,
            x='TotalCharge',
            kde=True);
```

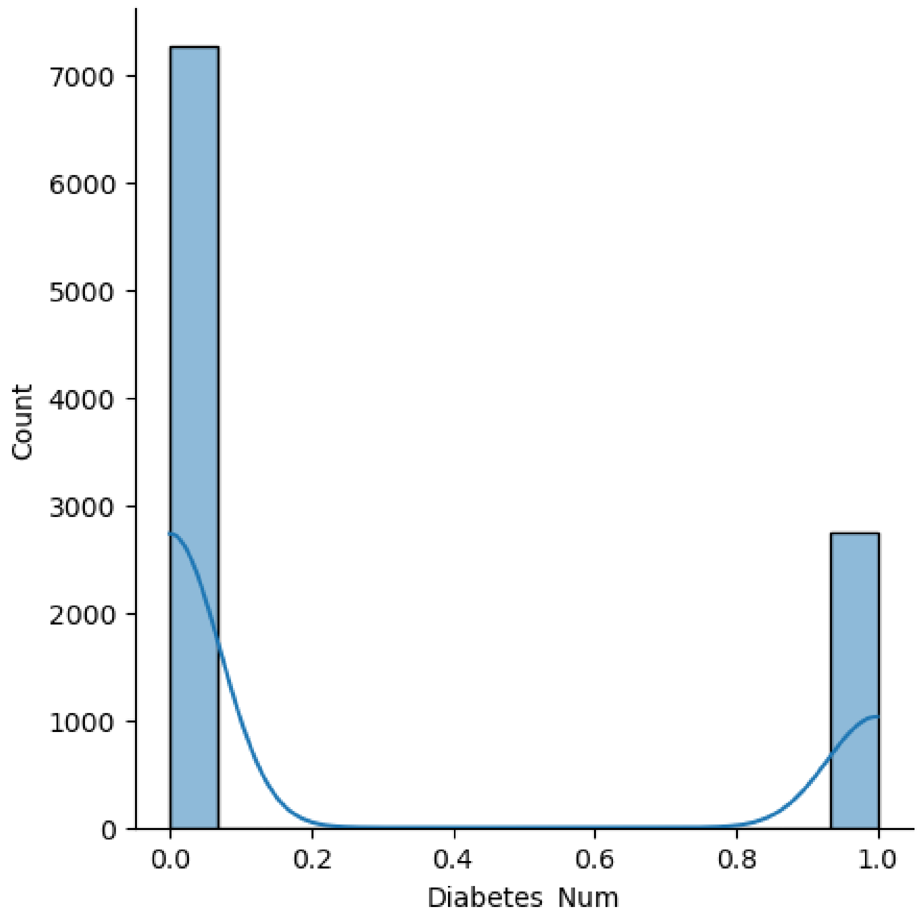


[20]:

```
# Distribution of Total Charge
sns.displot(data=pruned_df_num,
            x='Initial_days',
            kde=True);
```

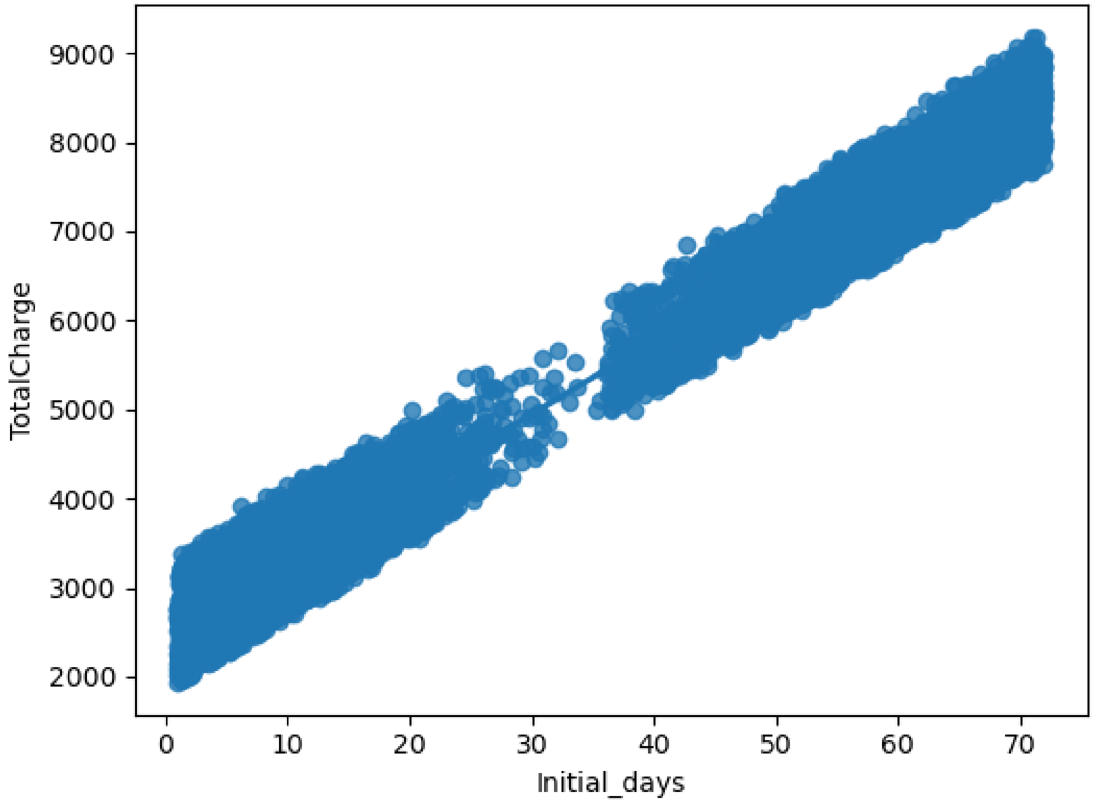


[21]:
```
# Distribution of Total Charge
sns.displot(data=pruned_df_num,
            x='Diabetes_Num',
            kde=True);
```

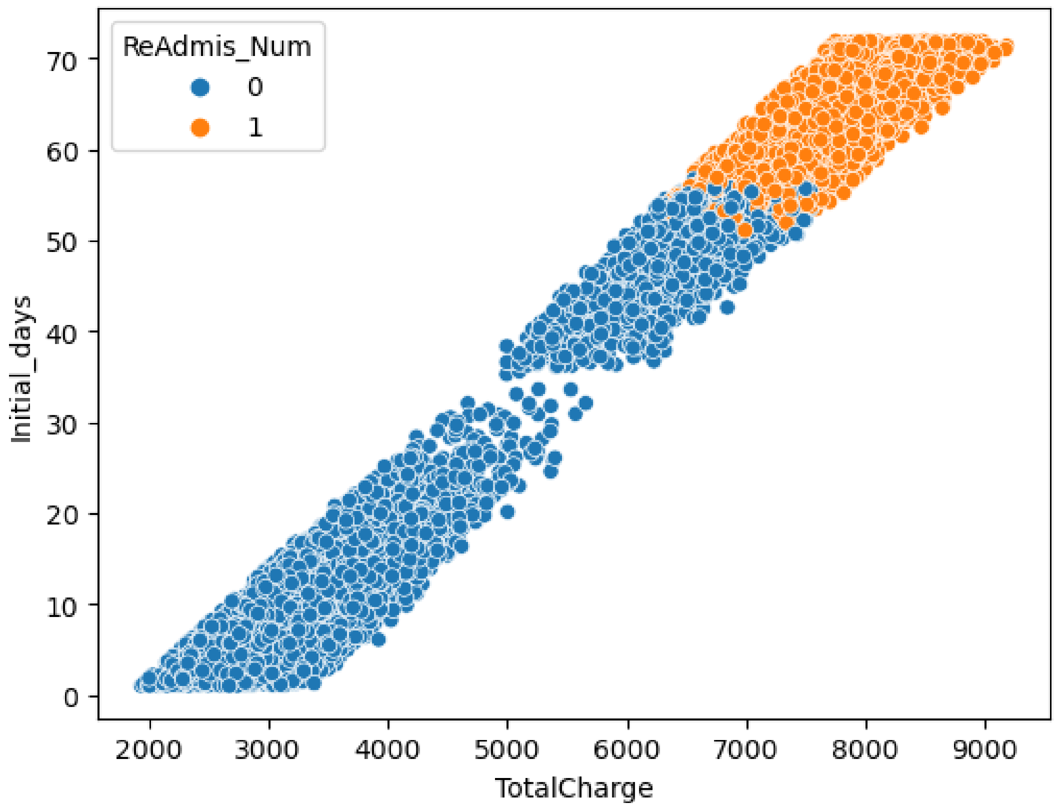


Bivariate Analysis

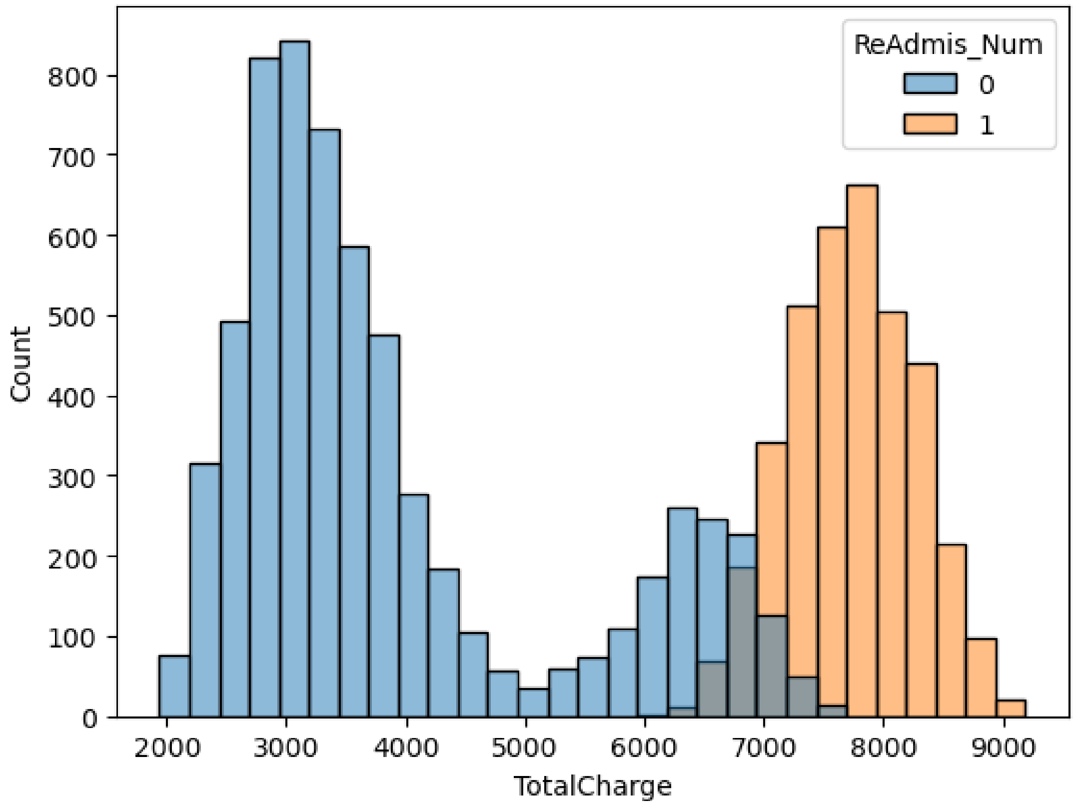
[22]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.regplot(x="Initial_days",
            y="TotalCharge",
            data=pruned_df_num,
            ci=None);
plt.show()
```



[23]:
```
sns.scatterplot(x='TotalCharge',
                y='Initial_days',
                data=pruned_df_num,
                hue='ReAdmis_Num');
```

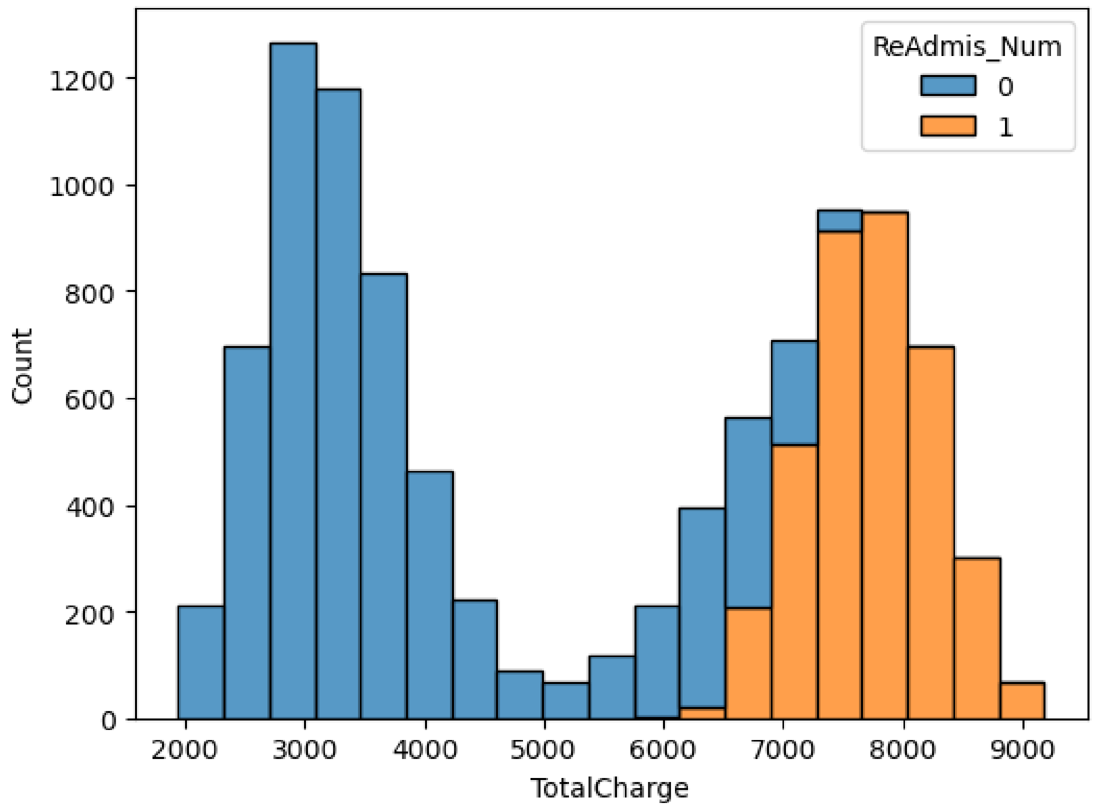


[24]:
```
sns.histplot(hue="ReAdmis_Num", x="TotalCharge", binwidth=250,␣
 ↪data=pruned_df_num);
plt.show()
```



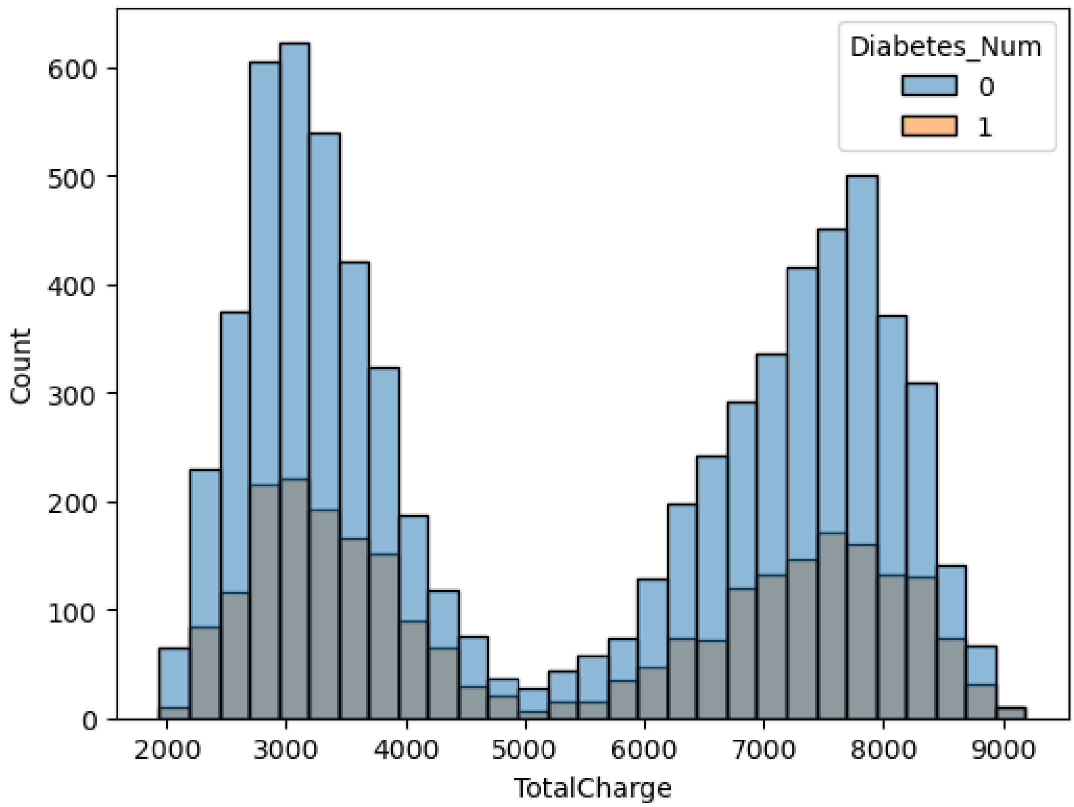
[25]:

```
sns.histplot(hue="ReAdmis_Num", x="TotalCharge", multiple="stack",␣
 ↪data=pruned_df_num);
plt.show()
```

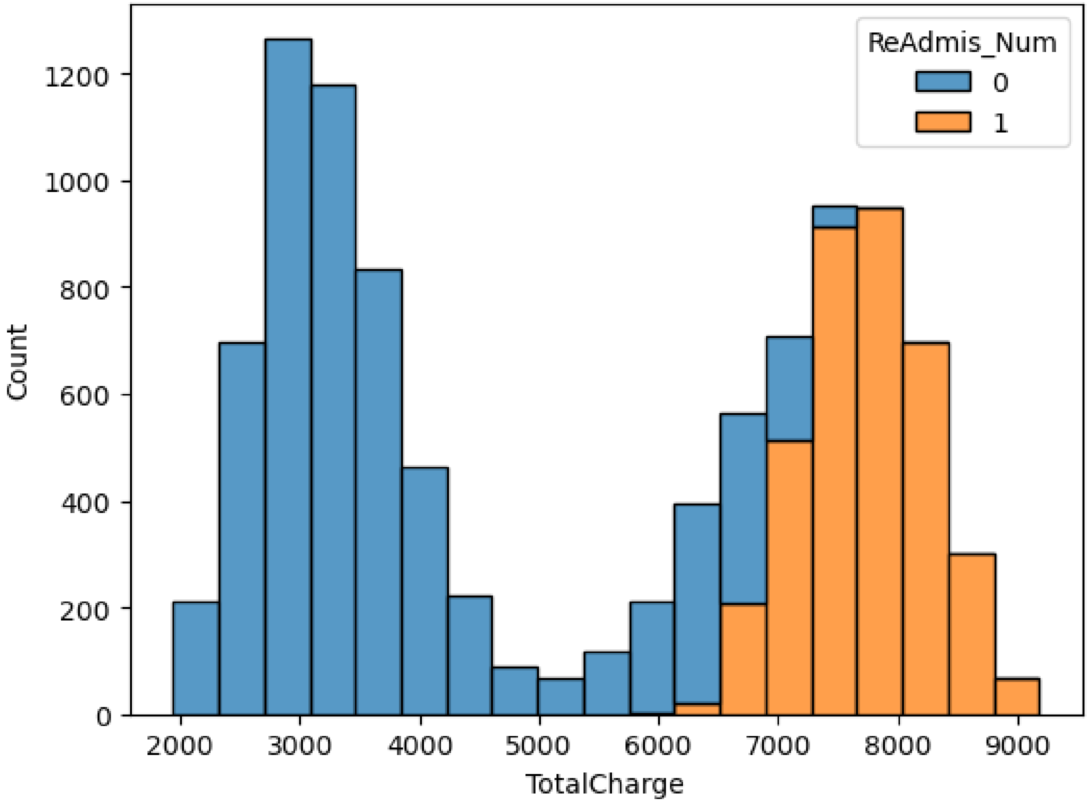


[26]:

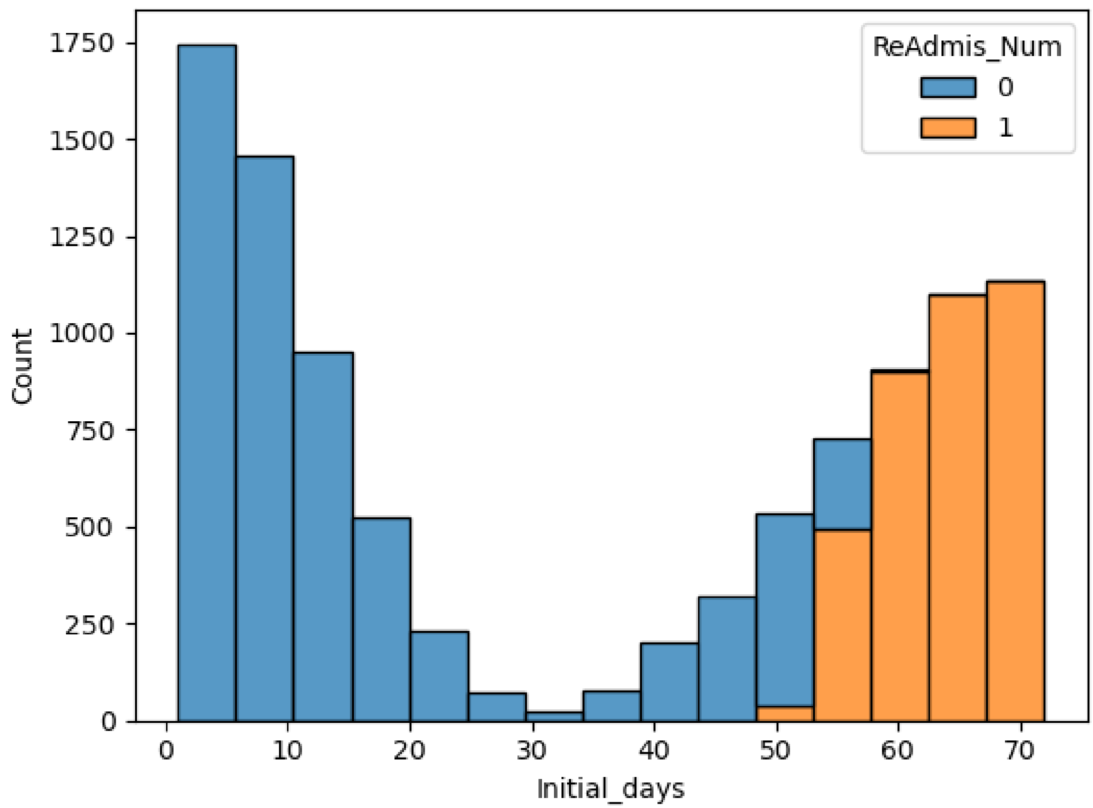
```
sns.histplot(hue="Diabetes_Num", x="TotalCharge", binwidth=250,␣
    ↪data=pruned_df_num);
plt.show()
```



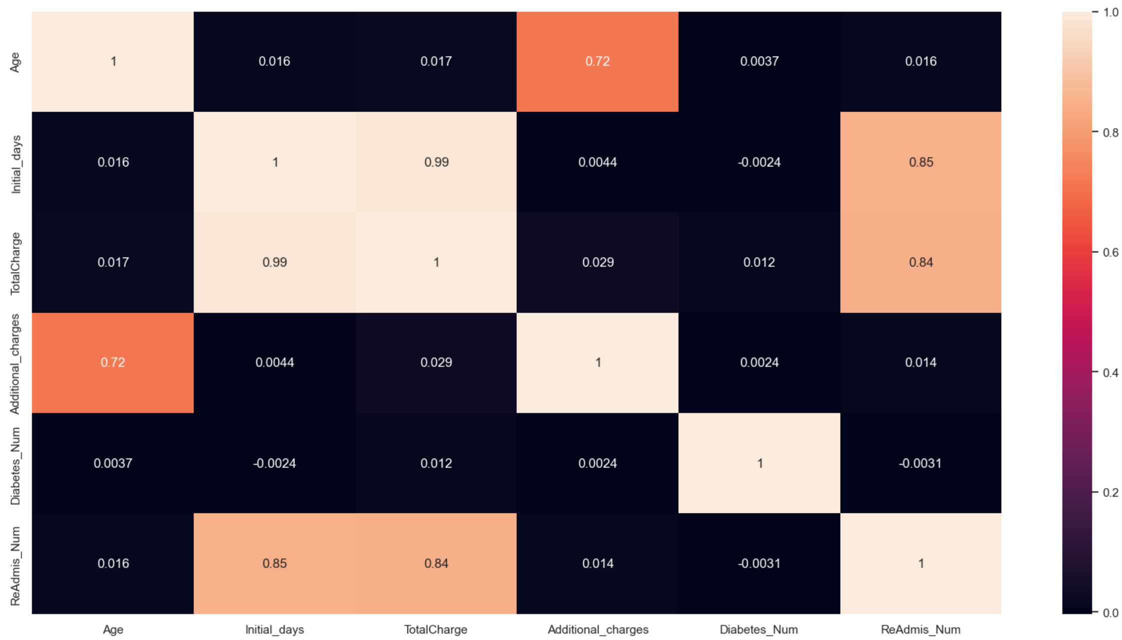
[27]:
```
sns.histplot(hue="ReAdmis_Num", x="TotalCharge", multiple="stack",␣
    ↪data=pruned_df_num);
plt.show()
```



[28]:
```
sns.histplot(hue="ReAdmis_Num", x="Initial_days", multiple="stack",␣
↪data=pruned_df_num);
plt.show()
```



[29]:
```
# 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':(20,10)})
sns.heatmap(pruned_df_num.corr(), annot=True);
```



0.3 Regression Model w/Most Terms - Compare to p-value

[30]:
```
# 90% Train, 10% Test
X = df_num.drop(['TotalCharge'], axis=1)

y = pruned_df_num['TotalCharge']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,␣
↪random_state=123)
```

[31]:
```
# Look into Stats Models w/p values
X_train2 = sm.add_constant(X_train)

model = sm.OLS(y_train,X_train2)

print(model.fit().summary())

p_vals = dict(model.fit().pvalues[1:])
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.977
Model:                            OLS   Adj. R-squared:                  0.977
Method:                 Least Squares   F-statistic:                 1.323e+04
Date:                Sun, 08 Jan 2023   Prob (F-statistic):               0.00
Time:                        15:57:29   Log-Likelihood:                -50607.
```

```
No. Observations:                7000   AIC:                         1.013e+05
Df Residuals:                    6977   BIC:                         1.014e+05
Df Model:                          22
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
const               2406.3630     71.153     33.820      0.000    2266.882
2545.844
CaseOrder              0.0019      0.002      0.750      0.453      -0.003
0.007
Zip                   -0.0005      0.000     -1.396      0.163      -0.001
0.000
Lat                   -1.3303      0.796     -1.670      0.095      -2.892
0.231
Lng                   -0.9506      0.633     -1.502      0.133      -2.191
0.290
Population             0.0002      0.000      0.658      0.511      -0.000
0.001
Children              -0.8629      1.859     -0.464      0.643      -4.507
2.781
Age                   -3.6349      0.278    -13.056      0.000      -4.181
-3.089
Income                -0.0001      0.000     -0.742      0.458      -0.000
0.000
VitD_levels           -0.2698      1.992     -0.135      0.892      -4.175
3.636
Doc_visits             3.6556      3.819      0.957      0.338      -3.831
11.142
Full_meals_eaten       2.0850      3.971      0.525      0.600      -5.700
9.870
vitD_supp              1.1346      6.390      0.178      0.859     -11.392
13.661
Initial_days          81.6749      0.273    298.824      0.000      81.139
82.211
Additional_charges     0.0166      0.001     18.969      0.000       0.015
0.018
Item1                  5.3760      5.779      0.930      0.352      -5.952
16.704
Item2                 -1.0377      5.316     -0.195      0.845     -11.459
9.383
Item3                  2.2263      4.904      0.454      0.650      -7.387
11.839
Item4                  3.4226      4.426      0.773      0.439      -5.253
12.099
```

```
Item5                  2.3406      4.595      0.509      0.610      -6.667
11.348
Item6                 -2.3712      4.751     -0.499      0.618     -11.684
6.942
Item7                  2.2815      4.444      0.513      0.608      -6.429
10.992
Item8                 -0.6209      4.226     -0.147      0.883      -8.906
7.664
==============================================================================
Omnibus:                      632.224   Durbin-Watson:                   1.978
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              212.550
Skew:                           0.140   Prob(JB):                     7.01e-47
Kurtosis:                       2.194   Cond. No.                     1.28e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.28e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Prune Based On P-Value and Hypothesis Question

[32]:
```python
# list comprehension --> Verify Fields < 0.05 p-value
[key for key in p_vals.keys() if p_vals[key] < 0.05]
```

[32]:
```
['Age', 'Initial_days', 'Additional_charges']
```

[33]:
```python
# pruned_df_num = pruned_df_num.drop(['Overweight_Num', 'Gender_Num'], axis=1)
pruned_df_num.columns
```

[33]:
```
Index(['Age', 'Initial_days', 'TotalCharge', 'Additional_charges',
       'Diabetes_Num', 'ReAdmis_Num'],
      dtype='object')
```

[34]:
```python
pruned_df_num.head()
```

[34]:
```
   Age  Initial_days  TotalCharge  Additional_charges  Diabetes_Num  \
0   53     10.585770  3726.702860        17939.403420             1
1   51     15.129562  4193.190458        17612.998120             0
2   53      4.772177  2434.234222        17505.192460             1
3   78      1.714879  2127.830423        12993.437350             0
4   22      1.254807  2113.073274         3716.525786             0

   ReAdmis_Num
0            0
1            0
2            0
```

```
3             0
4             0
```

Multiple Regression Model Run Again

[35]:
```
# 90% Train, 10% Test
# X = pruned_df_num.drop(['TotalCharge'], axis=1)
# USE Hypothesis Predictors
X = pruned_df_num[['Age', 'Initial_days', 'Additional_charges','ReAdmis_Num',␣
 ↪'Diabetes_Num']]

y = pruned_df_num['TotalCharge']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,␣
 ↪random_state=123)
```

[36]:
```
# Look into Stats Models w/p values
X_train2 = sm.add_constant(X_train)

model02 = sm.OLS(y_train,X_train2)

print(model02.fit().summary())

p_vals = dict(model02.fit().pvalues[1:])
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TotalCharge   R-squared:                       0.977
Model:                            OLS   Adj. R-squared:                  0.977
Method:                 Least Squares   F-statistic:                 5.902e+04
Date:                Sun, 08 Jan 2023   Prob (F-statistic):               0.00
Time:                        15:57:29   Log-Likelihood:                -50569.
No. Observations:                7000   AIC:                         1.012e+05
Df Residuals:                    6994   BIC:                         1.012e+05
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
const               2461.0510     12.722    193.453      0.000    2436.113
2485.989
Age                   -3.6174      0.276    -13.101      0.000      -4.159
-3.076
Initial_days          81.0227      0.290    279.243      0.000      80.454
81.591
```

```
Additional_charges      0.0165       0.001      19.028       0.000        0.015
0.018
ReAdmis_Num            51.2127      15.845       3.232       0.001       20.152
82.273
Diabetes_Num           77.9340       8.859       8.798       0.000       60.569
95.300
==============================================================================
Omnibus:                      673.792   Durbin-Watson:                   1.978
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              218.685
Skew:                           0.138   Prob(JB):                     3.26e-48
Kurtosis:                       2.180   Cond. No.                     5.98e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.98e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[37]:
```python
# list comprehension --> Verify Fields < 0.05 p-value
[key for key in p_vals.keys() if p_vals[key] < 0.05]
```

[37]:
```
['Age', 'Initial_days', 'Additional_charges', 'ReAdmis_Num', 'Diabetes_Num']
```

Train | Test | Split

[38]:
```python
import numpy as np
from sklearn.model_selection import train_test_split
```

[39]:
```python
pruned_df_num.columns
```

[39]:
```
Index(['Age', 'Initial_days', 'TotalCharge', 'Additional_charges',
       'Diabetes_Num', 'ReAdmis_Num'],
      dtype='object')
```

[40]:
```python
# X = pruned_df_num.drop('TotalCharge', axis=1) # Everything 'but'
# y = pruned_df_num['TotalCharge']
```

[41]:
```python
# 90% Train, 10% Test
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,␣
↪random_state=123)
```

[42]:
```python
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, y_train)
```

[42]:
```
LinearRegression()
```

[43]:
```
print(lm.intercept_)
```

```
2461.0510427290787
```

[44]:
```
# List coefficients relating to each feature in our dataset
coeff_df = pd.DataFrame(lm.coef_, X.columns, columns = ['Coefficient'])
coeff_df
```

[44]:

	Coefficient
Age	-3.617354
Initial_days	81.022694
Additional_charges	0.016543
ReAdmis_Num	51.212723
Diabetes_Num	77.934007

[45]:
```
lm.coef_
```

[45]:
```
array([-3.61735397e+00,  8.10226940e+01,  1.65428450e-02,  5.12127226e+01,
        7.79340070e+01])
```

[46]:
```
# Each coeficients from X_train above
X_train.columns
```

[46]:
```
Index(['Age', 'Initial_days', 'Additional_charges', 'ReAdmis_Num',
       'Diabetes_Num'],
      dtype='object')
```

0.4 Model Predictions

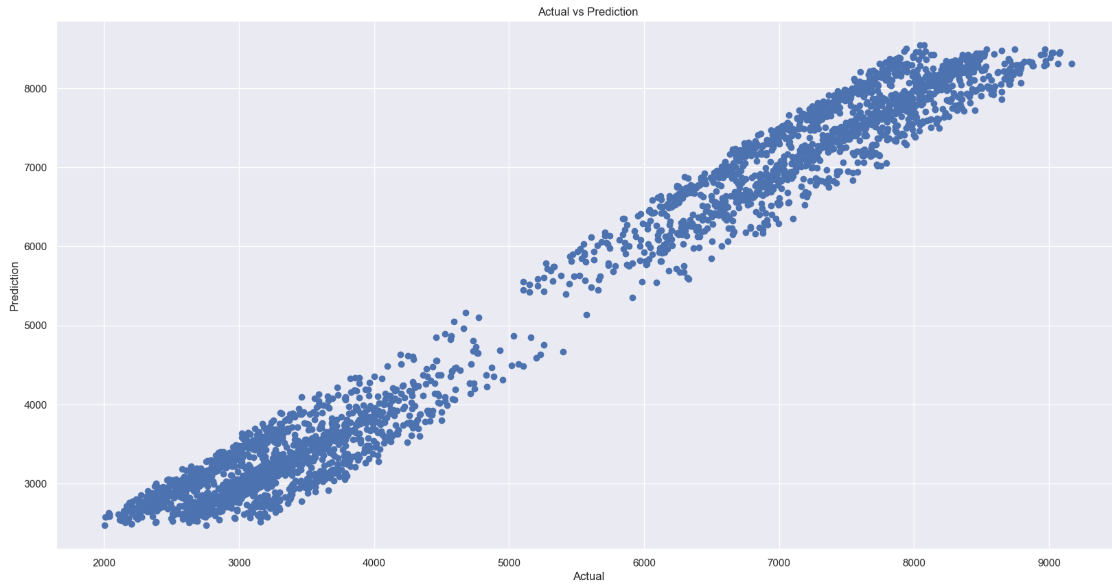
[47]:
```
predictions = lm.predict(X_test)
predictions
```

[47]:
```
array([3803.88627685, 3292.77655045, 7773.91401581, …, 7930.63809981,
       8434.66756507, 3113.39395265])
```

[48]:
```
# Measure of fit
lm.score(X_test, y_test)
```

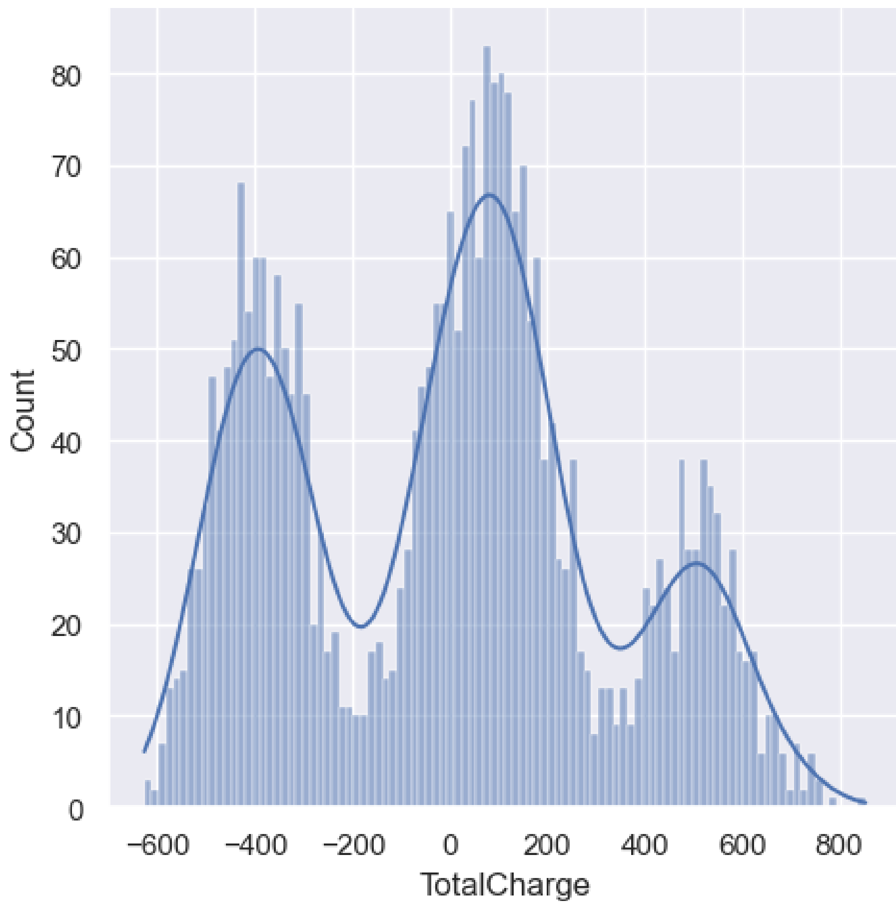
[48]:
```
0.9769194574255485
```

[49]:
```
plt.scatter(y_test, predictions);
plt.title("Actual vs Prediction")
plt.xlabel("Actual")
plt.ylabel("Prediction");
```



0.5 Regression Evaluation Metrics

[50]:
```
sns.displot(data=(y_test-predictions), bins=100, kde=True);
```



[51]:
```
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 271.844754051452
MSE: 109256.26995173047
RMSE: 330.5393621820713
```

[52]:
```
# Average Total Cost
df['TotalCharge'].mean()
```

[52]: 5312.1727687502

[53]:
```
# R squared
print('R Squared:', metrics.r2_score(y_test, predictions))
```

```
R Squared: 0.9769194574255485
```