D208 T1

October 2, 2021

[2]:
```
import pandas as pd
from sklearn import linear_model
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from scipy import stats
import numpy as np
```

[3]:
```
pd.options.mode.chained_assignment = None  # default='warn' ---- ignores false warning for database writing
```

[4]:
```
# write csv into datafile and select columns to analyze
df = pd.read_csv('medical_clean.csv')
df = df[['Children', 'Age', 'Income', 'Gender', 'ReAdmis', 'VitD_levels', 'Doc_visits', 'Complication_risk', 'Initial_days','Additional_charges']].copy()
print(df.head())
print(df.describe())
```

```
   Children  Age    Income  Gender ReAdmis  VitD_levels  Doc_visits  \
0         1   53  86575.93    Male      No    19.141466           6
1         3   51  46805.99  Female      No    18.940352           4
2         3   53  14370.14  Female      No    18.057507           4
3         0   78  39741.49    Male      No    16.576858           4
4         1   22   1209.56  Female      No    17.439069           5

  Complication_risk  Initial_days  Additional_charges
0            Medium     10.585770        17939.403420
1              High     15.129562        17612.998120
2            Medium      4.772177        17505.192460
3            Medium      1.714879        12993.437350
4               Low      1.254807         3716.525786
           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

       Initial_days  Additional_charges
count  10000.000000        10000.000000
mean      34.455299        12934.528587
std       26.309341         6542.601544
min        1.001981         3125.703000
25%        7.896215         7986.487755
50%       35.836244        11573.977735
75%       61.161020        15626.490000
max       71.981490        30566.070000
```

[5]:
```
# set male to 1 and others to 0 for categorical multivarate linear regression␣
 ↪(nominal categorical not ordinal)
di = {'Male': 1, 'Female': 0, 'Nonbinary': 0}
di2 = {'Yes': 1, 'No': 0}
di3 = {'High': 1, 'Medium': 0, 'Low':0}
df = df.replace({'Gender': di, 'ReAdmis': di2, 'Complication_risk': di3})
print(df.head())
```

```
   Children  Age     Income  Gender  ReAdmis  VitD_levels  Doc_visits  \
0         1   53   86575.93       1        0    19.141466           6
1         3   51   46805.99       0        0    18.940352           4
2         3   53   14370.14       0        0    18.057507           4
3         0   78   39741.49       1        0    16.576858           4
4         1   22    1209.56       0        0    17.439069           5

   Complication_risk  Initial_days  Additional_charges
0                  0     10.585770        17939.403420
1                  1     15.129562        17612.998120
2                  0      4.772177        17505.192460
3                  0      1.714879        12993.437350
4                  0      1.254807         3716.525786
```

[6]:
```
# check for duplicated and null values
print(df.loc[df.duplicated()])
print(df.isnull().sum())
```

```
Empty DataFrame
Columns: [Children, Age, Income, Gender, ReAdmis, VitD_levels, Doc_visits,
Complication_risk, Initial_days, Additional_charges]
Index: []
Children              0
Age                   0
Income                0
Gender                0
ReAdmis               0
```

```
VitD_levels           0
Doc_visits            0
Complication_risk     0
Initial_days          0
Additional_charges    0
dtype: int64
```

[7]:
```
# check for outliers
print(df.shape)
df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]
print(df.shape)
```

```
(10000, 10)
(9630, 10)
```

[8]:
```
reg = linear_model.LinearRegression()
reg.fit(df[['Children', 'Age', 'Income', 'Gender', 'ReAdmis', 'VitD_levels',␣
 ↪'Doc_visits', 'Initial_days', 'Complication_risk']], df.Additional_charges)
print(reg.coef_)
print(reg.intercept_)
```

```
[ 3.00064336e+01  2.27576501e+02 -1.25677894e-04  1.89890279e+02
  3.89974972e+02  5.46915028e+00  4.07437670e+01 -8.41696784e+00
  5.00486328e+02]
287.9138896222885
```

[9]:
```
test = sm.OLS(df['Additional_charges'], df[['Children', 'Age', 'Income',␣
 ↪'Gender', 'ReAdmis', 'VitD_levels', 'Doc_visits', 'Complication_risk',␣
 ↪'Initial_days']]).fit()
print(test.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:     Additional_charges   R-squared (uncentered):
0.901
Model:                            OLS   Adj. R-squared (uncentered):
0.901
Method:                 Least Squares   F-statistic:
9764.
Date:                Sat, 02 Oct 2021   Prob (F-statistic):
0.00
Time:                        23:12:44   Log-Likelihood:
-94785.
No. Observations:                9630   AIC:
1.896e+05
Df Residuals:                    9621   BIC:
1.897e+05
```

```
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
=====
                         coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----
Children              31.2054     24.269      1.286      0.199     -16.366
78.777
Age                  227.8640      2.195    103.803      0.000     223.561
232.167
Income              2.852e-05      0.002      0.016      0.987      -0.004
0.004
Gender               195.6127     92.438      2.116      0.034      14.416
376.810
ReAdmis              381.6689    183.238      2.083      0.037      22.484
740.854
VitD_levels           16.4535     13.143      1.252      0.211      -9.310
42.217
Doc_visits            51.7115     40.265      1.284      0.199     -27.216
130.639
Complication_risk    503.9649     98.137      5.135      0.000     311.596
696.334
Initial_days          -8.1656      3.338     -2.446      0.014     -14.709
-1.622
==============================================================================
Omnibus:                   247241.951   Durbin-Watson:                   1.988
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              996.450
Skew:                           0.409   Prob(JB):                    4.20e-217
Kurtosis:                       1.654   Cond. No.                     1.83e+05
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 1.83e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[10]:
```
# unvarite analysis histogram
df.hist()
plt.savefig('hospital_pyplot.jpg')
plt.tight_layout()
plt.close()
```

[11]:
```
# bivariate analysis heatmap
sns.heatmap(df.corr(), annot=True)
plt.savefig('heatmap.jpg', bbox_inches='tight')
plt.close()
```

[12]:
```
#regression of targeted variables
reg2 = linear_model.LinearRegression()
reg2.fit(df[['Age', 'ReAdmis', 'Initial_days']], df.Additional_charges)
print(reg2.coef_)
print(reg2.intercept_)
```

```
[227.65158239 400.55511529  -8.57817249]
900.3142458728853
```

[13]:
```
#residual plot
ax1 = df.plot(kind = 'scatter', x = 'Additional_charges', y = 'Age', color = 'r')
plt.savefig('scatter1.jpg')
plt.close()
ax2 = df.plot(kind = 'scatter', x = 'Additional_charges', y = 'ReAdmis', color =␣
↪'b')
plt.savefig('scatter2.jpg')
plt.close()
ax3 = df.plot(kind = 'scatter', x = 'Additional_charges', y = 'Initial_days',␣
↪color = 'g')
plt.savefig('scatter3.jpg')
plt.close()
```

[14]:
```
scatterTest = sm.OLS(df['Additional_charges'], df[['Age', 'ReAdmis',␣
↪'Initial_days']]).fit()
print(scatterTest.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:     Additional_charges   R-squared (uncentered):
0.901
Model:                            OLS   Adj. R-squared (uncentered):
0.901
Method:                 Least Squares   F-statistic:
2.907e+04
Date:                Sat, 02 Oct 2021   Prob (F-statistic):
0.00
Time:                        23:15:38   Log-Likelihood:
-94820.
No. Observations:                9630   AIC:
1.896e+05
Df Residuals:                    9627   BIC:
```

```
1.897e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Age            239.1196      1.282    186.507      0.000     236.606     241.633
ReAdmis        150.7741    179.872      0.838      0.402    -201.813     503.361
Initial_days    -0.2494      3.096     -0.081      0.936      -6.318       5.819
==============================================================================
Omnibus:                    91110.113   Durbin-Watson:                   1.986
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              938.124
Skew:                           0.306   Prob(JB):                    1.95e-204
Kurtosis:                       1.599   Cond. No.                        261.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[15]:
```
df.to_csv('cleaned_data.csv')
```

[ ]:
```

```