Linear Regression using Python

Almira A. Legaspina

March 10, 2022

Import Libraries

In [77]:
```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
import numpy as np
```

Problem Narrative

```
- Indian Liver Patient
    Want to determine what age mostly who has low in total protiens
```

Here is the Indian Liver Patient Dataset

In [78]:
```
data = pd.read_excel('Indian Liver Patient Dataset (ILPD).xlsx', names=['age','gender','TB','DB','Alkphos',
                                                                        'Sgpt','Sgot','TP','ALB','A/G','selectorField'])
```

In [79]:
```
data.head()
```

Out[79]:

	age	gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	A/G	selectorField
0	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
1	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
2	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
3	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
4	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	1

In [80]:
```
#check the shape of the dataframe
data.shape
```

Out[80]: (582, 11)

In [81]:
```
data.tail()
```

Out[81]:

	age	gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	A/G	selectorField
577	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	2
578	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1
579	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
580	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
581	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	2

```
data.describe()
```

	age	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	A/G	selectorField
count	582.000000	582.000000	582.000000	582.000000	582.000000	582.000000	582.000000	582.000000	578.000000	582.000000
mean	44.711340	3.303265	1.488488	290.754296	80.824742	110.068729	6.482646	3.141581	0.947145	1.286942
std	16.181921	6.213926	2.810324	243.108929	182.757696	289.141876	1.086306	0.796176	0.319863	0.452723
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000	1.000000
25%	33.000000	0.800000	0.200000	175.250000	23.000000	25.000000	5.800000	2.600000	0.700000	1.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.940000	1.000000
75%	57.750000	2.600000	1.300000	298.000000	60.750000	87.000000	7.200000	3.800000	1.100000	2.000000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000	2.000000

```
#define a function called "plot_boxplot"
def plot_boxplot(data,ft):
    data.boxplot(column=[ft])
    plt.grid(False)
    plt.show()
```

In [84]:
```
#define a function called "outliers" which returns a list of index of outliers
#IQR = Q3 - Q1
#+/-1.5*IQR

def outliers(data,ft):
    Q1 = data[ft].quantile(0.25)
    Q3 = data[ft].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    ls = data.index[ (data[ft] < lower_bound) | (data[ft] > upper_bound)]

    return ls
```

In [85]:
```
#create a function an empty list to store the output indices from multiple columns

index_list =[]
for col in ['TB','DB','Alkphos','Sgpt','Sgot','TP','ALB','A/G']:
    index_list.extend(outliers(data,col))
```

```
cleanedata = remove(data, index_list)
```

```
cleanedata.shape
```

```
(400, 11)
```

```
cleanedata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 400 entries, 2 to 581
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   age            400 non-null    int64
 1   gender         400 non-null    object
 2   TB             400 non-null    float64
 3   DB             400 non-null    float64
 4   Alkphos        400 non-null    int64
 5   Sgpt           400 non-null    int64
 6   Sgot           400 non-null    int64
 7   TP             400 non-null    float64
 8   ALB            400 non-null    float64
 9   A/G            396 non-null    float64
 10  selectorField  400 non-null    int64
dtypes: float64(5), int64(5), object(1)
memory usage: 53.7+ KB
```

```
x = cleanedata['age']
```

In [111]:
```
x = cleanedata['age']
y = cleanedata['TP']
```

Data Cleaned

In [112]:
```
x
```

Out[112]:
```
2      58
3      72
4      46
5      26
6      29
      ..
570    90
578    40
579    52
580    31
581    38
Name: age, Length: 400, dtype: int64
```

In [113]:
```
y
```

Out[113]:
```
2      6.8
3      7.3
4      7.6
5      7.0
6      6.7
      ...
570    6.9
578    6.0
579    6.4
580    6.8
581    7.3
```

In [113]:
```
y
```

Out[113]:
```
2      6.8
3      7.3
4      7.6
5      7.0
6      6.7
      ...
570    6.9
578    6.0
579    6.4
580    6.8
581    7.3
Name: TP, Length: 400, dtype: float64
```

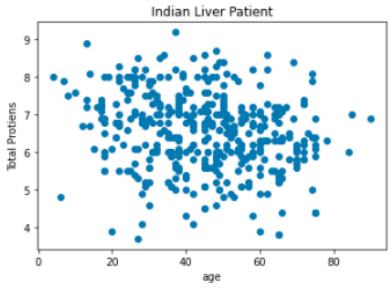
In [114]:
```
x2 =sm.add_constant(x)
model =sm.OLS(y,x2)
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                     TP   R-squared:                       0.040
Model:                            OLS   Adj. R-squared:                  0.038
Method:                 Least Squares   F-statistic:                     16.56
Date:                Thu, 10 Mar 2022   Prob (F-statistic):           5.68e-05
Time:                        23:20:38   Log-Likelihood:                -571.95
No. Observations:                 400   AIC:                             1148.
Df Residuals:                     398   BIC:                             1156.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          7.0791      0.146     48.593      0.000       6.793       7.366
age           -0.0124      0.003     -4.070      0.000      -0.018      -0.006
==============================================================================
Omnibus:                        3.802   Durbin-Watson:                   1.463
Prob(Omnibus):                  0.149   Jarque-Bera (JB):                3.887
Skew:                          -0.227   Prob(JB):                        0.143
Kurtosis:                       2.838   Cond. No.                         137.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

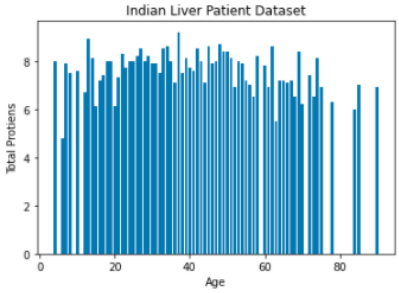
2. Graphs

In [96]:
```
# scatter the age and totalProtiens
plt.scatter(x,y)
# set a title and labels
plt.title('Indian Liver Patient')
plt.xlabel('age')
plt.ylabel('Total Protiens')
plt.show()
```

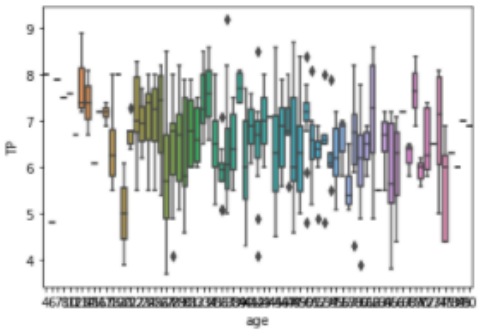


In [98]:
```
plt.bar(x,y)
# set title and labels
plt.title('Indian Liver Patient Dataset')
plt.xlabel('Age')
plt.ylabel('Total Protiens')
plt.show()
```

In [98]:
```
plt.bar(x,y)
# set title and labels
plt.title('Indian Liver Patient Dataset')
plt.xlabel('Age')
plt.ylabel('Total Protiens')
plt.show()
```



In [105]:
```
sns.boxplot(x, y)
sns.title('Indian Liver Patient Dataset')
sns.xlabel('Age')
sns.ylabel('Total Protiens')
sns.show()
```



Steps taken to solve the problem

- Cleaned the dataset
- get variable x and variable y
- regression results
- make a graph to support

Present the model

Analytical Model