Predicting a firm's investment: Panel Data Models

The analysis will attempt to predict a firm's level of investment given its market value, capital stock, and 20 annual observations on 11 firms.

Number of observations - 220 (20 years for 11 firms) Number of variables - 5 Variables name definitions: Invest - Gross investment in dollars in the year 1947 Value - Market value as of Dec. 31 in dollar in the year 1947 Capital - Stock of plant and equipment in dollars as of 1947 Firm - General Motors, US Steel, General Electric, Chrysler, Atlantic Refining, IBM, Union Oil, Westinghouse, Goodyear, Diamond Match, American Steel Year - 1935 - 1954

In [1]:
```
# Load Modules and Functions
import statsmodels.api as sm
import statsmodels as sms
import seaborn as sns
import statsmodels.formula.api as smf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import linearmodels as plm
import numpy as np
import linearmodels as plm
import scipy.stats as stats
```

Part 2

Descriptive analysis of data

In [60]:
```
# Load the data:
df = pd.read_csv("/Users/snehilshandiliya/Desktop/Grunfeld.csv")
df
```

0	1	317.600	3078.500	2.800	General Motors	1935
1	2	391.800	4661.700	52.600	General Motors	1936
2	3	410.600	5387.100	156.900	General Motors	1937
3	4	257.700	2792.200	209.200	General Motors	1938
4	5	330.800	4313.200	203.400	General Motors	1939
...	...	...	...	...	...	...
215	216	4.770	36.494	75.847	American Steel	1950
216	217	6.532	46.082	77.367	American Steel	1951
217	218	7.329	57.616	78.631	American Steel	1952
218	219	9.020	57.441	80.215	American Steel	1953
219	220	6.281	47.165	83.788	American Steel	1954

220 rows × 6 columns

Number of observations - 220 (20 years for 11 firms)

Number of variables - 5

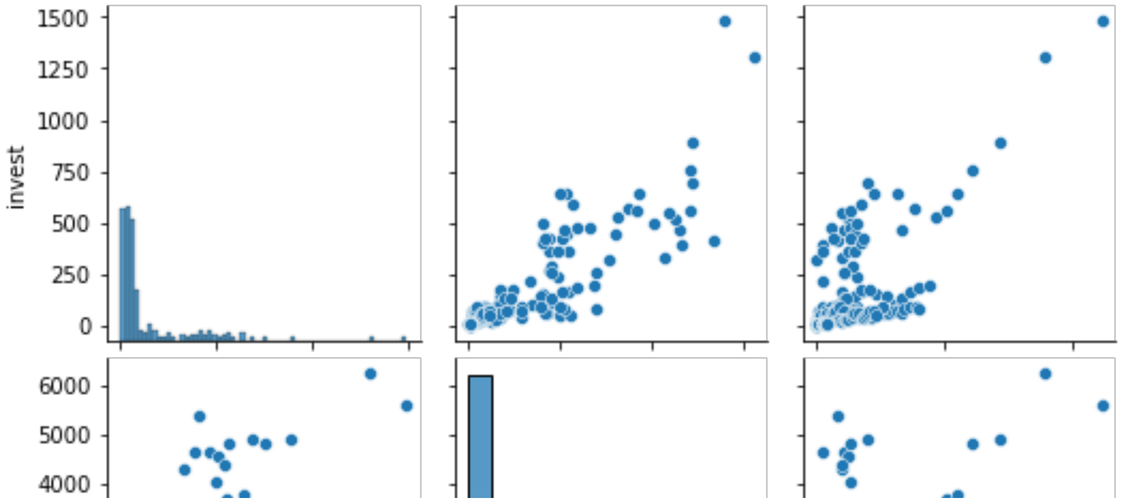
Variables name definitions:

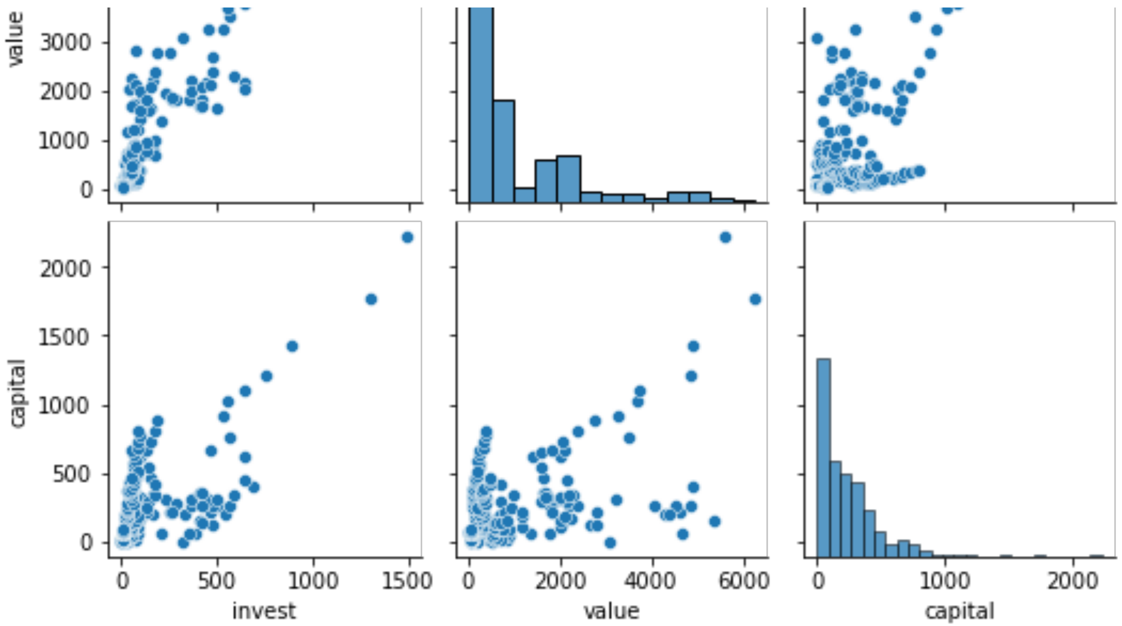
```
invest  - Gross investment in 1947 dollars
value   - Market value as of Dec. 31 in 1947 dollars
capital - Stock of plant and equipment in 1947 dollars
firm    - General Motors, US Steel, General Electric, Chrysler,
          Atlantic Refining, IBM, Union Oil, Westinghouse, Goodyear,
          Diamond Match, American Steel
year    - 1935 - 1954
```

In [3]:

```python
import seaborn as sns
sns.pairplot(df,vars=['invest', 'value','capital'])
```

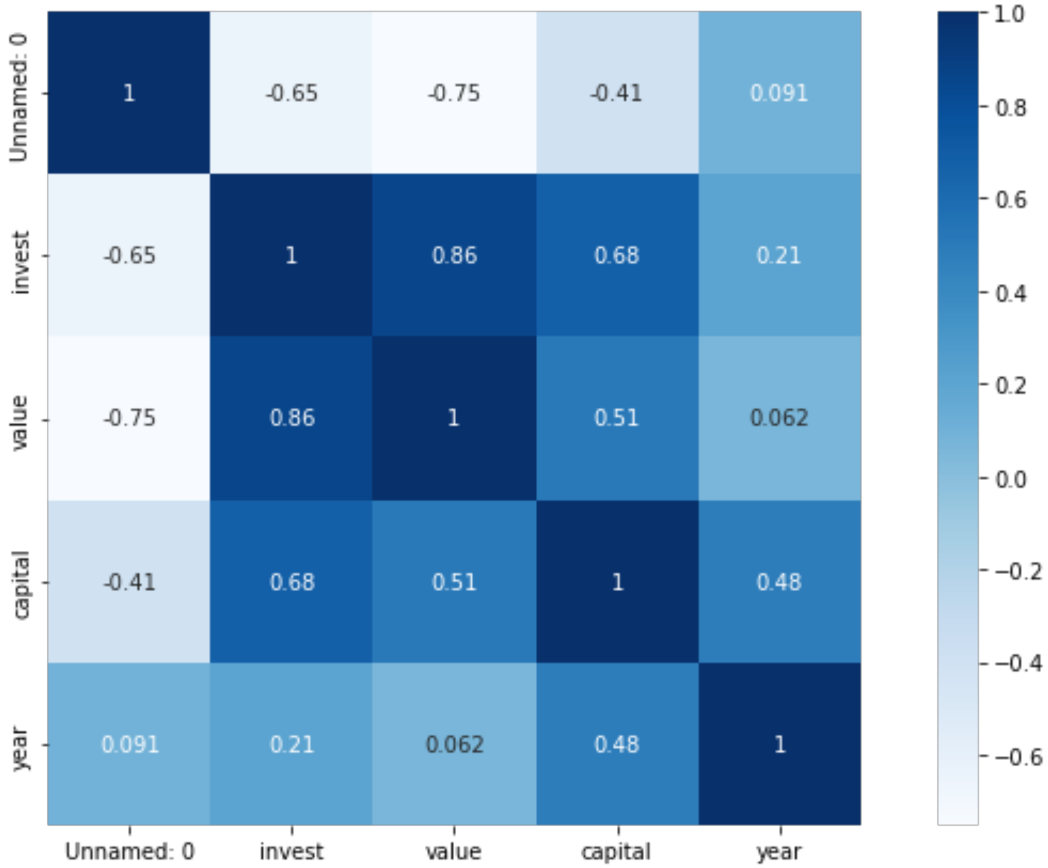
Out[3]: <seaborn.axisgrid.PairGrid at 0x7ff1f5132c40>




value
3000
2000
1000
0
capital
2000
1500
1000
500
0
0
500
1000
1500
invest
0
2000
4000
6000
value
0
1000
2000
capital

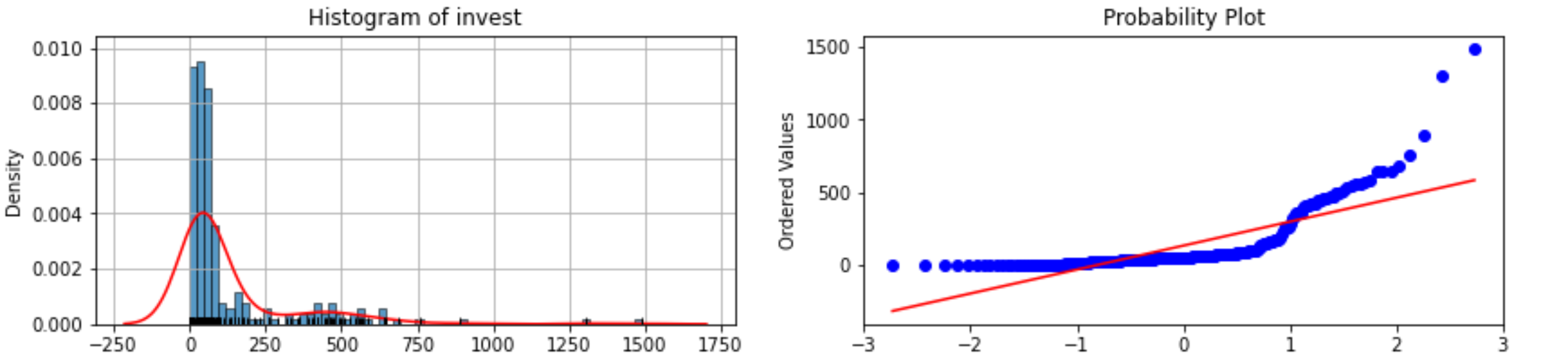
In [4]:
```
plt.figure(figsize=(13,7))
c= df.corr()
sns.heatmap(c,cmap="Blues",annot=True,square = True)
```

Out[4]: <AxesSubplot:>



In [5]:

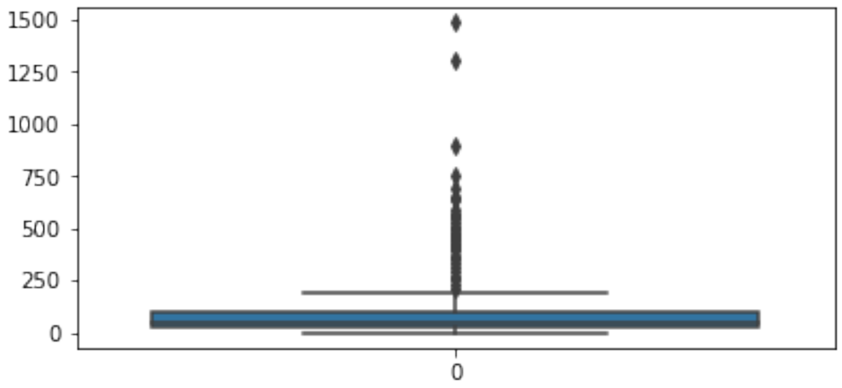
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of invest")
sns.histplot(df.invest, stat = "density")
sns.kdeplot(df.invest, color = "red")
sns.rugplot(df.invest, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df.invest, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df['invest'])
```



invest

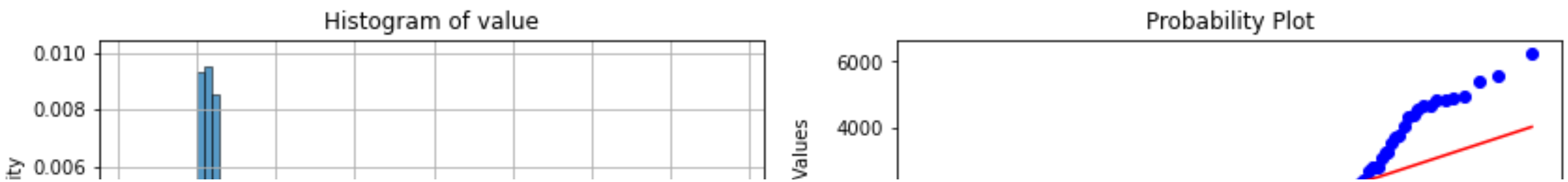
Theoretical quantiles

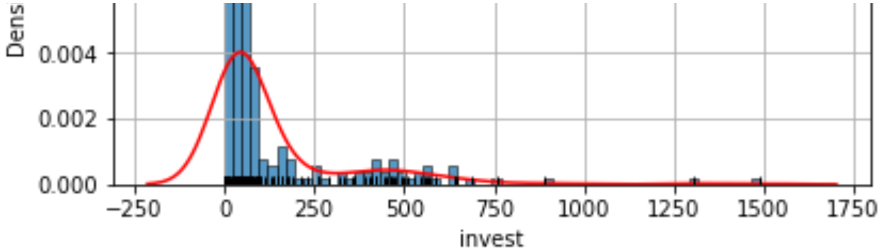
Out[5]: <AxesSubplot:>

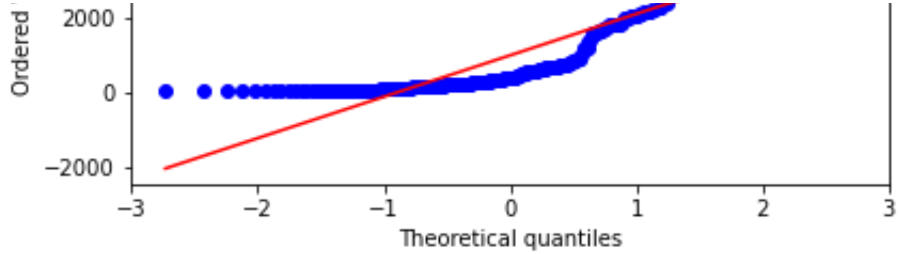


In [6]:

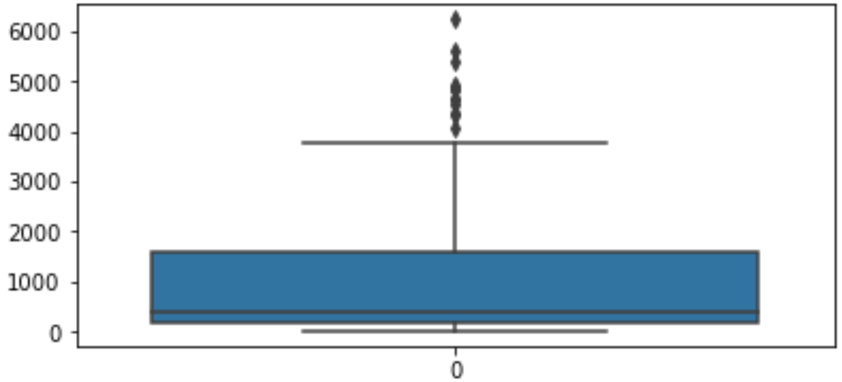
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of value")
sns.histplot(df.invest, stat = "density")
sns.kdeplot(df.invest, color = "red")
sns.rugplot(df.invest, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df.value, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df['value'])

plt.subplot(3,2,4)

plt.scatter(df["value"],df["invest"])
plt.xlabel("value")
plt.ylabel("invest")
plt.title("Realtionship between total investment and valuation ")
```

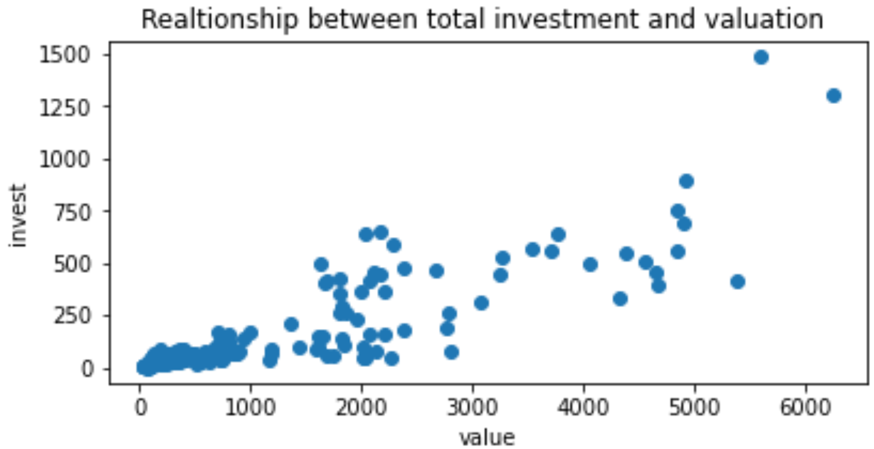






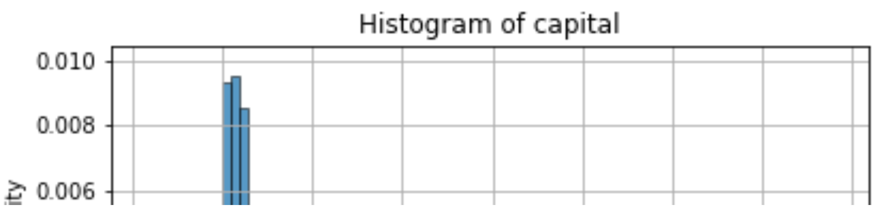
Out[6]: Text(0.5, 1.0, 'Realtionship between total investment and valuation ')

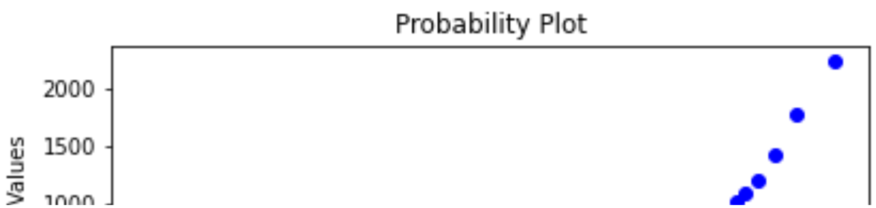


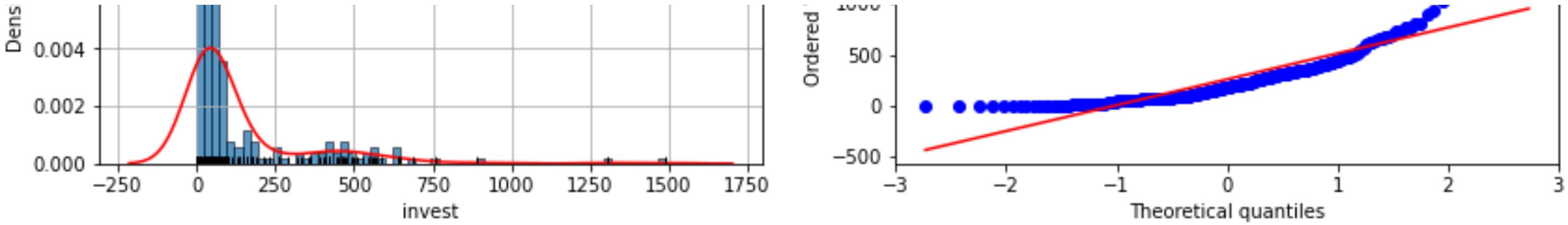


In [7]:

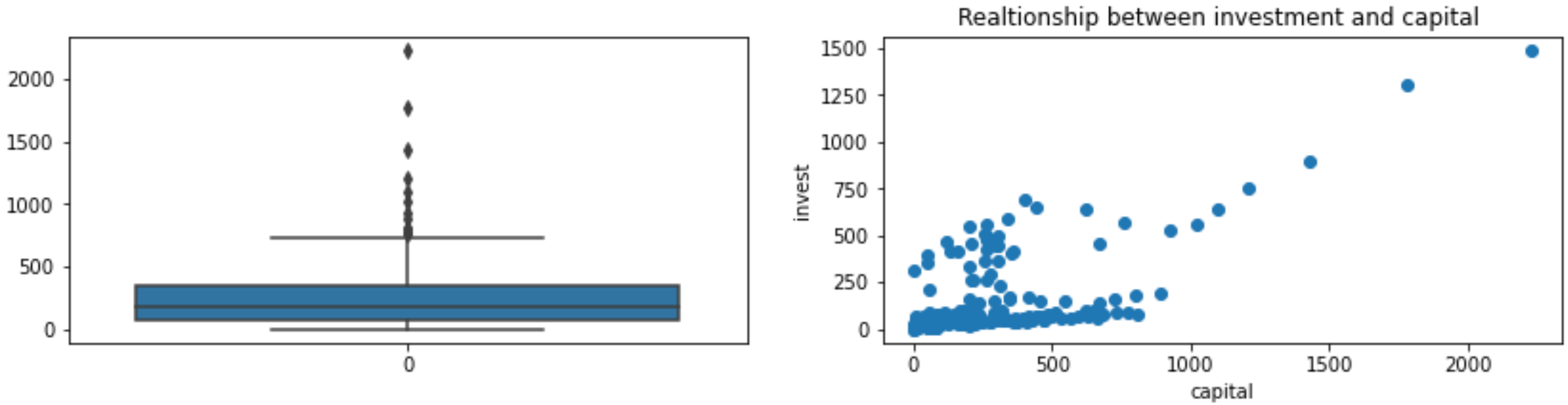
```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of capital")
sns.histplot(df.invest, stat = "density")
sns.kdeplot(df.invest, color = "red")
sns.rugplot(df.invest, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df.capital, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df['capital'])

plt.subplot(3,2,4)

plt.scatter(df["capital"],df["invest"])
plt.xlabel("capital")
plt.ylabel("invest")
plt.title("Realtionship between investment and capital ")
```







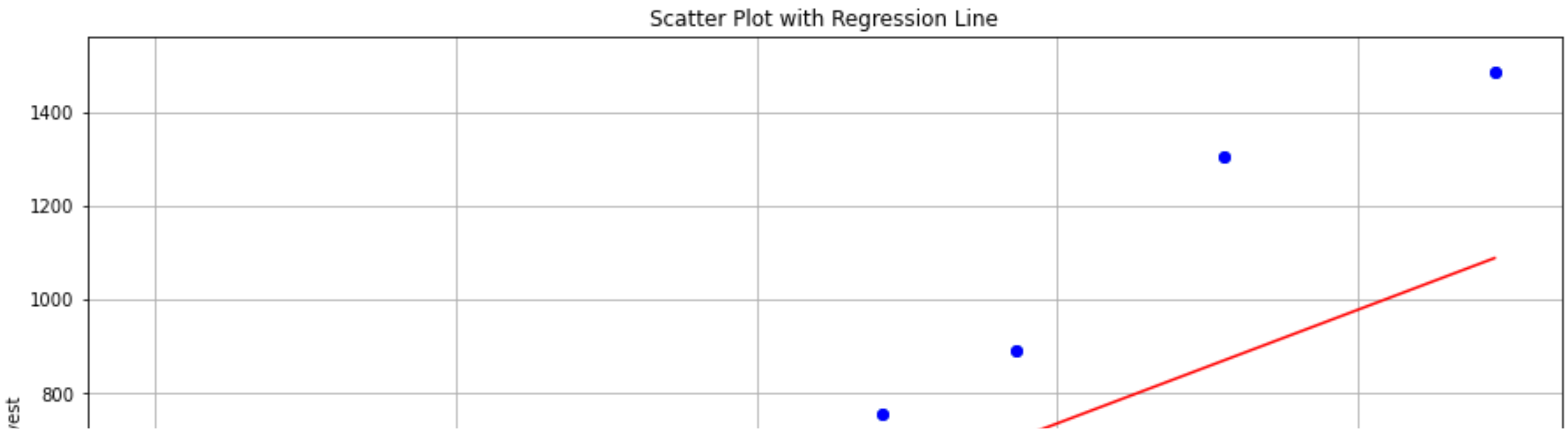
Out[7]: Text(0.5, 1.0, 'Realtionship between investment and capital ')

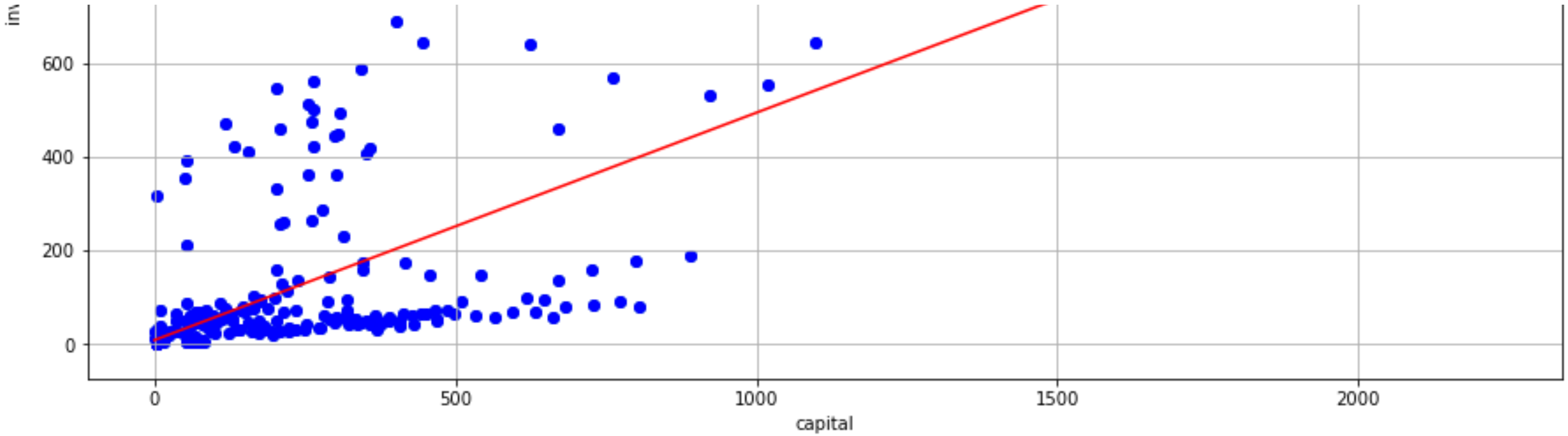


In [63]:

```
plt.figure(figsize = (15, 8))


plt.scatter(df["capital"], df["invest"])

# Create regression line
m,b = np.polyfit(df["capital"], df["invest"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.capital.max(), 1000)

# combining the two plots
plt.scatter(df["capital"], df["invest"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("invest")
plt.xlabel("capital")
plt.grid()
```

```
The slope of the regression line is: 0.48519136672262414 The Intercept is: 8.565055640258556
```

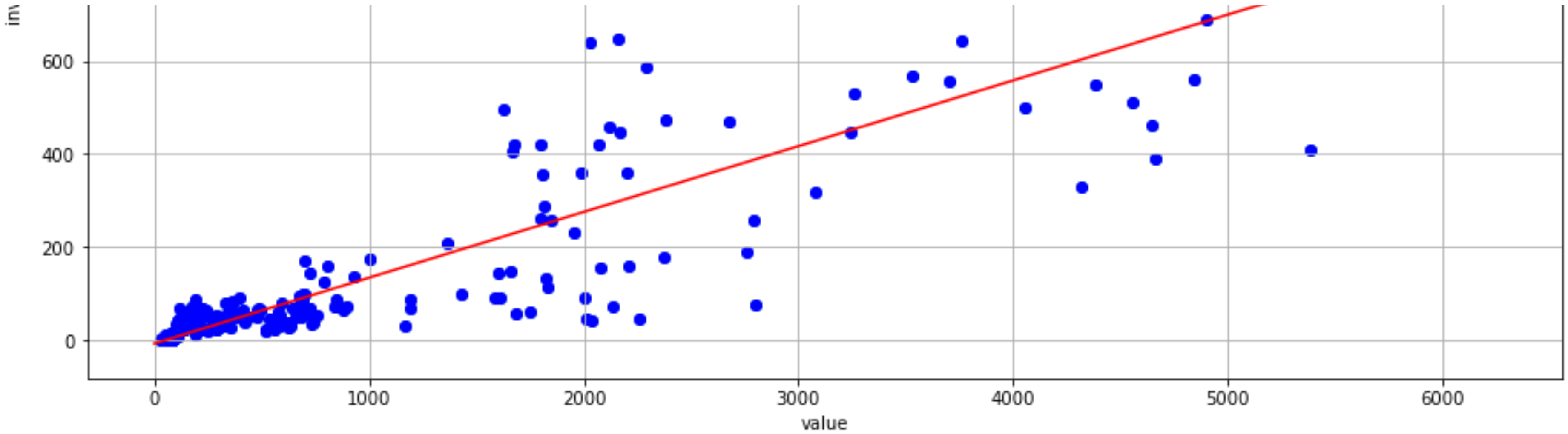




In [64]:

```
plt.figure(figsize = (15, 8))


plt.scatter(df["value"], df["invest"])

# Create regression line
m,b = np.polyfit(df["value"], df["invest"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.value.max(), 100)

# combining the two plots
plt.scatter(df["value"], df["invest"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("invest")
plt.xlabel("value")
plt.grid()
```

```
The slope of the regression line is: 0.141092580112949 The Intercept is: -6.169093085712734
```



In [10]:
```
# Specify the Model
model = smf.ols(formula='invest ~ capital + value + C(year)', data=df)

result = model.fit()

print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 invest   R-squared:                       0.822
Model:                            OLS   Adj. R-squared:                  0.803
Method:                 Least Squares   F-statistic:                     43.55
Date:                Wed, 30 Nov 2022   Prob (F-statistic):           1.27e-62
Time:                        15:33:04   Log-Likelihood:                -1298.8
No. Observations:                 220   AIC:                             2642.
Df Residuals:                     198   BIC:                             2716.
Df Model:                          21
Covariance Type:            nonrobust
=================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
```

```
------------------------------------------------------------------------------------
Intercept           -21.6815     28.354     -0.765      0.445     -77.597      34.234
C(year)[T.1936]     -15.1865     39.884     -0.381      0.704     -93.839      63.466
C(year)[T.1937]     -30.8415     39.958     -0.772      0.441    -109.640      47.957
C(year)[T.1938]     -25.9640     39.882     -0.651      0.516    -104.611      52.683
C(year)[T.1939]     -51.2476     39.902     -1.284      0.201    -129.936      27.441
C(year)[T.1940]     -27.5208     39.911     -0.690      0.491    -106.226      51.184
C(year)[T.1941]      -2.0012     39.928     -0.050      0.960     -80.739      76.737
C(year)[T.1942]      -0.3563     39.990     -0.009      0.993     -79.216      78.504
C(year)[T.1943]     -18.7958     39.997     -0.470      0.639     -97.671      60.079
C(year)[T.1944]     -19.4973     39.991     -0.488      0.626     -98.360      59.366
C(year)[T.1945]     -29.7423     40.002     -0.744      0.458    -108.627      49.142
C(year)[T.1946]      -6.1207     40.033     -0.153      0.879     -85.066      72.825
C(year)[T.1947]      -4.3649     40.312     -0.108      0.914     -83.860      75.130
C(year)[T.1948]      -2.8025     40.508     -0.069      0.945     -82.686      77.081
C(year)[T.1949]     -25.2951     40.683     -0.622      0.535    -105.522      54.932
C(year)[T.1950]     -24.9390     40.767     -0.612      0.541    -105.332      55.454
C(year)[T.1951]      -9.4694     40.792     -0.232      0.817     -89.912      70.973
C(year)[T.1952]      -3.8273     41.134     -0.093      0.926     -84.944      77.289
C(year)[T.1953]       4.0537     41.589      0.097      0.922     -77.961      86.068
C(year)[T.1954]      -9.3916     42.268     -0.222      0.824     -92.744      73.961
capital               0.2166      0.030      7.244      0.000       0.158       0.276
value                 0.1158      0.006     19.434      0.000       0.104       0.128
==============================================================================
Omnibus:                       33.290   Durbin-Watson:                   0.341
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              134.793
Skew:                           0.482   Prob(JB):                     5.37e-30
Kurtosis:                       6.711   Cond. No.                     3.42e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.42e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [11]:
```
## variables are not multicollinear
import statsmodels.stats.outliers_influence as smo
import patsy as pt

# extract matrices using patsy:
y, X = pt.dmatrices('invest ~ capital + value', data=df, return_type='dataframe')

# get VIF:
K = X.shape[1]
VIF = np.empty(K)
for i in range(K):
    VIF[i] = smo.variance_inflation_factor(X.values, i)
print(f'VIF: \n{VIF}\n')
# Refering to the threshold of 4 we can see that there exists almost no milticollinearity between the va
```

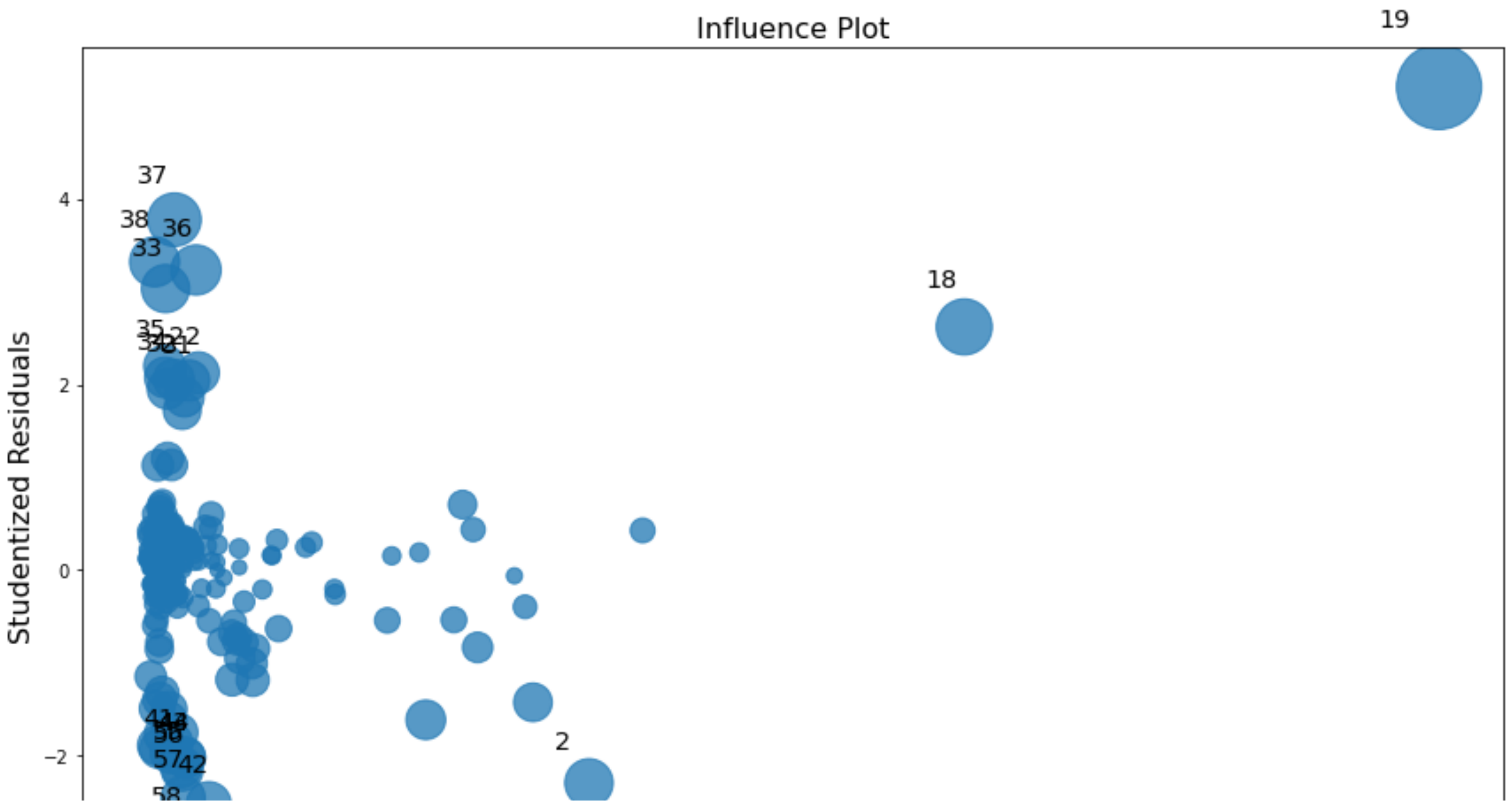
```
VIF: 
[1.9106169  1.35615621 1.35615621]
```

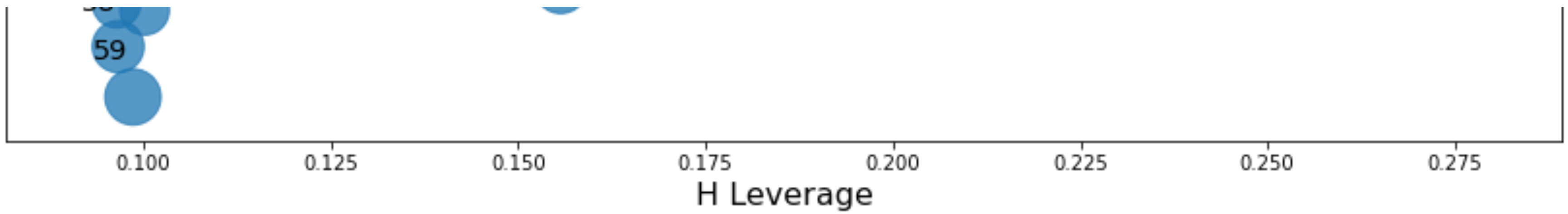
In [12]:
```
# Model Misspecification
import statsmodels.regression.linear_model as rg
import statsmodels.stats.diagnostic as dg

test =  dg.linear_reset(result, power=2,  test_type='fitted', use_f = True)
test
# Fail to reject Ho, therefore, the model seems to be correctly specified at order 2.
```

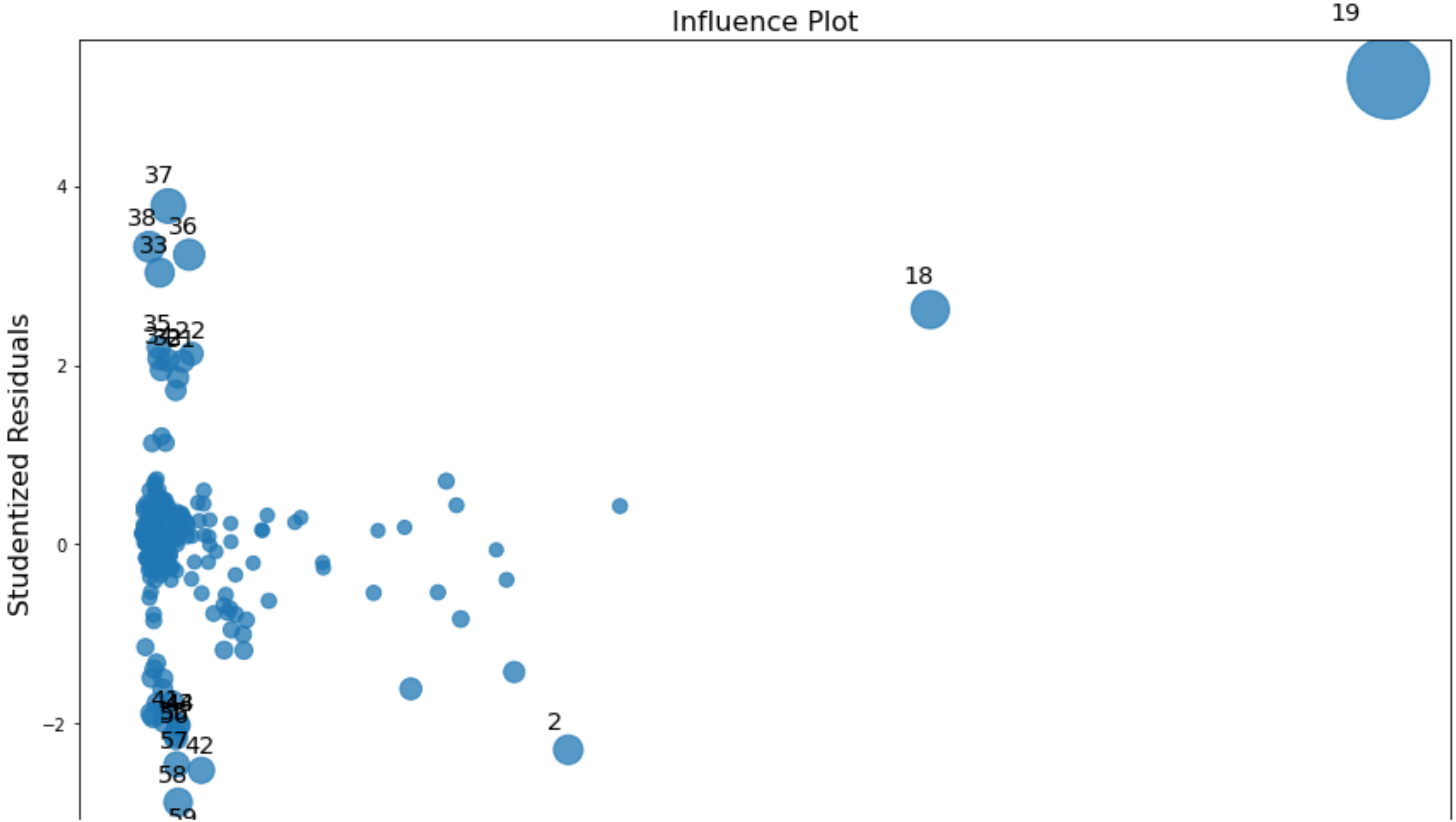
```
/Users/snehilshandilya/opt/anaconda3/lib/python3.9/site-packages/statsmodels/stats/diagnostic.py:1081: FutureWarning: Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be r
emoved in a future version.  Convert to a numpy array before indexing instead.
  aug = res.fittedvalues[:, None]
```

Out[12]:
```
<class 'statsmodels.stats.contrast.ContrastResults'>
<F test: F=36.95838532080777, p=6.1473891031121235e-09, df_denom=197, df_num=1>
```

In [13]:
```
# Outliers, high leverage, influential obs
figd, ax = plt.subplots(figsize=(12,8))
figd = sm.graphics.influence_plot(result, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sm.graphics.influence_plot(result, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```



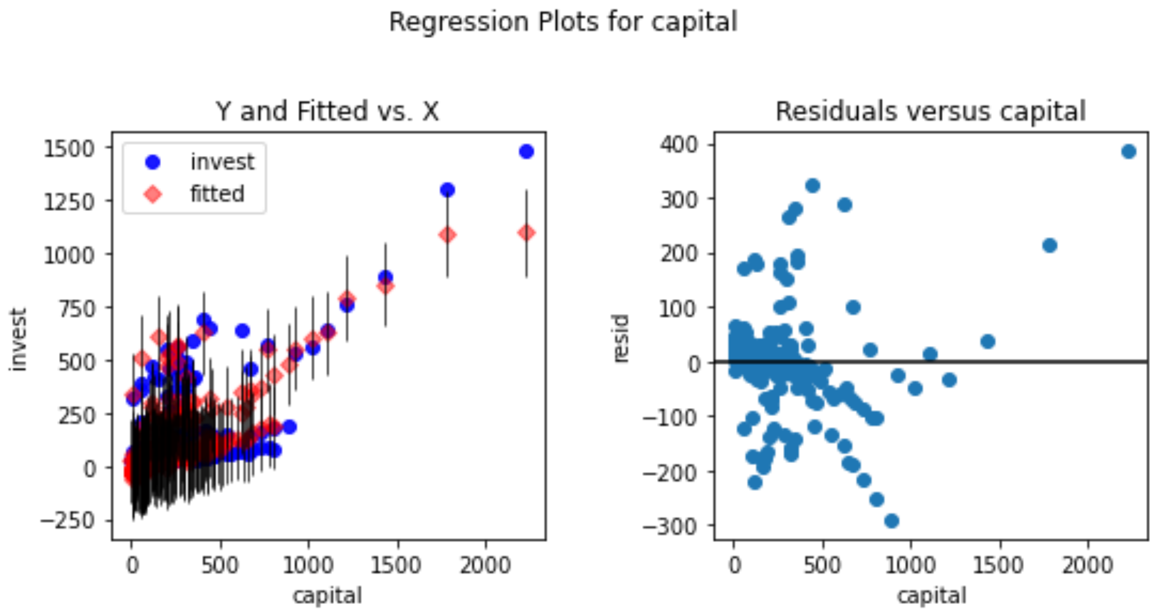

59
0.100
0.125
0.150
0.175
0.200
0.225
0.250
0.275
H Leverage

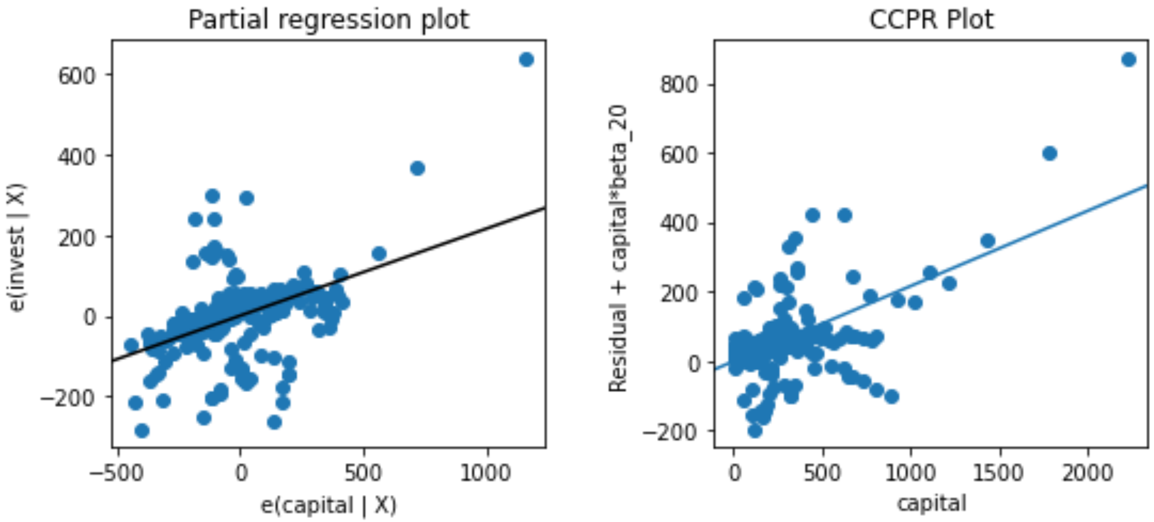

Influence Plot
19
37
38
36
33
18
22
2
57
42
58
4
2
0
−2
Studentized Residuals

In [14]:

```python
fig = sm.graphics.plot_regress_exog(result, "capital")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

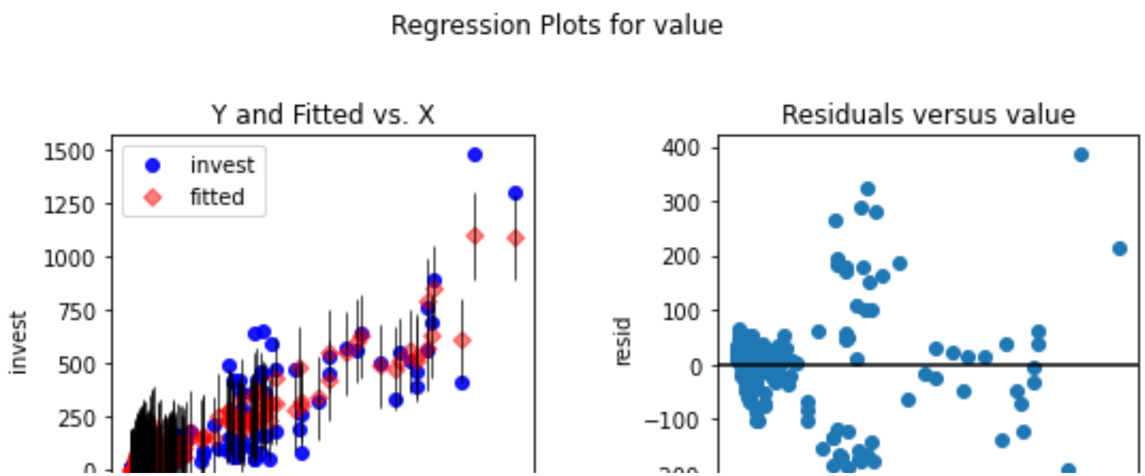
```
eval_env: 1
```

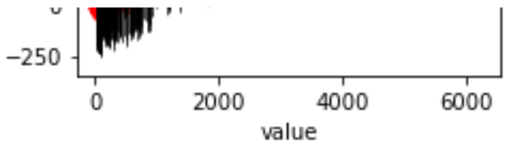


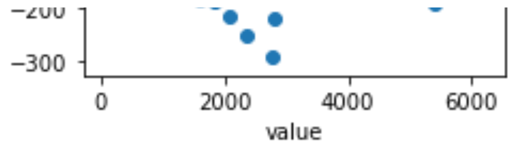


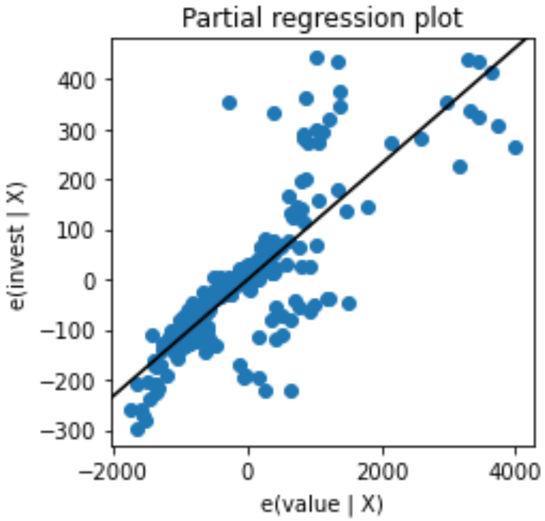
In [15]:
```
fig = sm.graphics.plot_regress_exog(result, "value")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

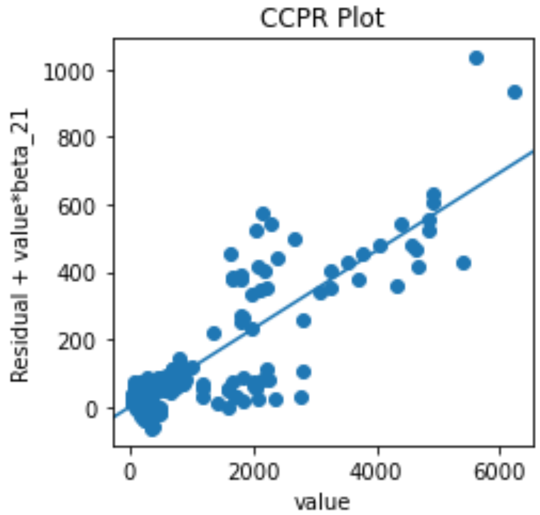
```
eval_env: 1
```











Part 3

Model preference

In [17]:
```
df = df.set_index(['firm', 'year'], drop = False)
```

In [18]:
```
## Pooled Effect model
dd= plm.PooledOLS.from_formula(formula='invest ~ capital + value', data=df)
```

```
dd= plm.PooledOLS.from_formula(formula='invest ~ capital + value', data=df)
results_ols = dd.fit()
## Random Effect model
reg_re = plm.RandomEffects.from_formula(
    formula='invest~ capital + value + C(year)', data=df)
results_re = reg_re.fit()
# Fixed effect model
reg_fe = plm.PanelOLS.from_formula(
    formula='invest~ capital + value + C(year) + EntityEffects', data=df)
results_fe = reg_fe.fit()

# print results:
theta_hat = results_re.theta.iloc[0, 0]
print(f'theta_hat: {theta_hat}\n')

table_ols = pd.DataFrame({'b': round(results_ols.params, 4),
                          'se': round(results_ols.std_errors, 4),
                          't': round(results_ols.tstats, 4),
                          'pval': round(results_ols.pvalues, 4)})
print(f'table_ols: \n{table_ols}\n')

table_re = pd.DataFrame({'b': round(results_re.params, 4),
                         'se': round(results_re.std_errors, 4),
                         't': round(results_re.tstats, 4),
                         'pval': round(results_re.pvalues, 4)})
print(f'table_re: \n{table_re}\n')

table_fe = pd.DataFrame({'b': round(results_fe.params, 4),
                         'se': round(results_fe.std_errors, 4),
                         't': round(results_fe.tstats, 4),
                         'pval': round(results_fe.pvalues, 4)})
print(f'table_fe: \n{table_fe}\n')
```

```
theta_hat: 0.0
```

```
table_ols:
              b      se        t  pval
capital  0.1824  0.0231   7.8938   0.0
value    0.1078  0.0056  19.4085   0.0

table_re:
                       b       se        t    pval
C(year)[T.1935] -21.6815  28.3544  -0.7647  0.4454
C(year)[T.1936] -36.8679  28.6125  -1.2885  0.1991
C(year)[T.1937] -52.5230  28.8556  -1.8202  0.0702
C(year)[T.1938] -47.6455  28.4664  -1.6737  0.0958
C(year)[T.1939] -72.9290  28.6168  -2.5485  0.0116
C(year)[T.1940] -49.2023  28.6555  -1.7170  0.0875
C(year)[T.1941] -23.6826  28.6487  -0.8267  0.4094
C(year)[T.1942] -22.0378  28.6157  -0.7701  0.4421
C(year)[T.1943] -40.4773  28.6857  -1.4111  0.1598
C(year)[T.1944] -41.1787  28.6946  -1.4351  0.1528
C(year)[T.1945] -51.4238  28.7672  -1.7876  0.0754
C(year)[T.1946] -27.8022  28.8404  -0.9640  0.3362
C(year)[T.1947] -26.0464  29.0588  -0.8963  0.3712
C(year)[T.1948] -24.4840  29.3033  -0.8355  0.4044
C(year)[T.1949] -46.9765  29.5447  -1.5900  0.1134
C(year)[T.1950] -46.6205  29.6854  -1.5705  0.1179
C(year)[T.1951] -31.1508  29.8330  -1.0442  0.2977
C(year)[T.1952] -25.5088  30.3064  -0.8417  0.4010
C(year)[T.1953] -17.6278  31.0110  -0.5684  0.5704
C(year)[T.1954] -31.0731  31.8745  -0.9749  0.3308
capital           0.2166   0.0299   7.2436  0.0000
value             0.1158   0.0060  19.4340  0.0000

table_fe:
                        b       se        t    pval
C(year)[T.1935]  -30.5344  16.9435  -1.8021  0.0731
C(year)[T.1936]  -47.4937  19.3696  -2.4520  0.0151
```

```
C(year)[T.1937]  -66.9101  21.5220  -3.1089  0.0022
C(year)[T.1938]  -66.1582  17.7154  -3.7345  0.0002
C(year)[T.1939]  -93.6338  19.2782  -4.8570  0.0000
C(year)[T.1940]  -70.3592  19.6594  -3.5789  0.0004
C(year)[T.1941]  -47.0222  19.3042  -2.4359  0.0158
C(year)[T.1942]  -48.5338  17.9305  -2.7068  0.0074
C(year)[T.1943]  -68.3069  18.6788  -3.6569  0.0003
C(year)[T.1944]  -68.8545  18.8876  -3.6455  0.0003
C(year)[T.1945]  -80.0739  19.6781  -4.0692  0.0001
C(year)[T.1946]  -58.2888  20.1604  -2.8913  0.0043
C(year)[T.1947]  -65.4120  18.4042  -3.5542  0.0005
C(year)[T.1948]  -68.8652  18.3640  -3.7500  0.0002
C(year)[T.1949]  -95.7352  18.6200  -5.1415  0.0000
C(year)[T.1950]  -97.9222  19.0366  -5.1439  0.0000
C(year)[T.1951]  -85.3691  20.5665  -4.1509  0.0001
C(year)[T.1952]  -87.0235  21.1280  -4.1189  0.0001
C(year)[T.1953]  -89.0470  23.0464  -3.8638  0.0002
C(year)[T.1954] -112.3284  23.1984  -4.8421  0.0000
capital            0.3514   0.0210  16.6964  0.0000
value              0.1167   0.0129   9.0219  0.0000
```

Hausman Test

In [19]:

```
import numpy.linalg
## Fixed
b_fe = results_fe.params
b_fe_cov = results_fe.cov

## Random
results_re = reg_re.fit()
b_re = results_re.params
b_re_cov = results_re.cov

# Hausman test of FE vs. RE
# (I) find overlapping coefficients:
common_coef = set(results_fe.params.index).intersection(results_re.params.index)

# (II) calculate differences between FE and RE:
b_diff = np.array(results_fe.params[common_coef] - results_re.params[common_coef])
df = len(b_diff)
b_diff.reshape((df, 1))
b_cov_diff = np.array(b_fe_cov.loc[common_coef, common_coef] - b_re_cov.loc[common_coef, common_coef])
b_cov_diff.reshape((df, df))

# (III) calculate test statistic:
stat = abs(np.transpose(b_diff) @ np.linalg.inv(b_cov_diff) @ b_diff)
pval = 1 - stats.chi2.cdf(stat, df)

print(f'stat: {stat}\n')
print(f'pval: {pval}\n')
```

```
stat: 39.99799754432716

pval: 0.010817543732512536


/var/folders/sv/s309_3dd79s_59j12prhgcd00000gn/T/ipykernel_5676/1767762092.py:16: FutureWarning: Passin
g a set as an indexer is deprecated and will raise in a future version. Use a list instead.
  b_diff = np.array(results_fe.params[common_coef] - results_re.params[common_coef])
```

```
/var/folders/sv/s309_3dd79s_59j12prhgcd00000gn/T/ipykernel_5676/1767762092.py:16: FutureWarning: Passin
g a set as an indexer is deprecated and will raise in a future version. Use a list instead.
  b_diff = np.array(results_fe.params[common_coef] - results_re.params[common_coef])
/var/folders/sv/s309_3dd79s_59j12prhgcd00000gn/T/ipykernel_5676/1767762092.py:19: FutureWarning: Passin
g a set as an indexer is deprecated and will raise in a future version. Use a list instead.
  b_cov_diff = np.array(b_fe_cov.loc[common_coef, common_coef] - b_re_cov.loc[common_coef, common_coef]
)

/var/folders/sv/s309_3dd79s_59j12prhgcd00000gn/T/ipykernel_5676/1767762092.py:19: FutureWarning: Passin
g a set as an indexer is deprecated and will raise in a future version. Use a list instead.
  b_cov_diff = np.array(b_fe_cov.loc[common_coef, common_coef] - b_re_cov.loc[common_coef, common_coef]
)
```

For Hausman test - Ho: preferred model is the Random Effects model Ha: Fixed Effects model is better. With low p-value we can see that we reject the null and conclude that the Fixed effect model is the better model.

Question 2

Qualitative Dependent Variable Models

Part 1

Economics/Finance question we are answering - Stevie

In [21]:
```
df1 = pd.read_csv("/Users/snehilshandilya/Desktop/framingham.csv")
df1
```

Out[21]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	he
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	1	0	313.0	179.0	92.0	25.97	
4234	1	51	3.0	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19.71	
4235	0	48	2.0	1	20.0	NaN	0	0	0	248.0	131.0	72.0	22.00	
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	

4238 rows × 16 columns

In [39]:
```
df1.isnull().any()
```

Out[39]:
```
male               False
age                False
education           True
currentSmoker      False
cigsPerDay          True
BPMeds              True
prevalentStroke    False
prevalentHyp       False
diabetes           False
totChol             True
sysBP              False
diaBP              False
BMI                 True
heartRate           True
glucose             True
TenYearCHD         False
dtype: bool
```

In [41]:
```
df2 = df1.fillna(df1.median())
df2
```

Out[41]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	he
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	1	0	313.0	179.0	92.0	25.97	
4234	1	51	3.0	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19.71	
4235	0	48	2.0	1	20.0	0.0	0	0	0	248.0	131.0	72.0	22.00	
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	

4238 rows × 16 columns

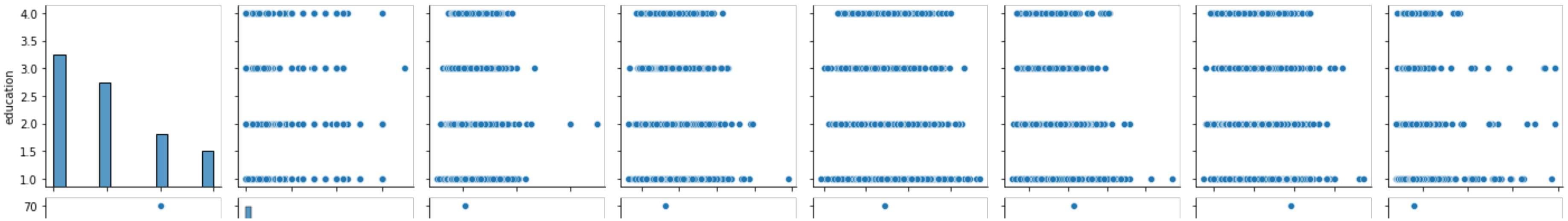
In [43]:
```
df2.isnull().any()
```

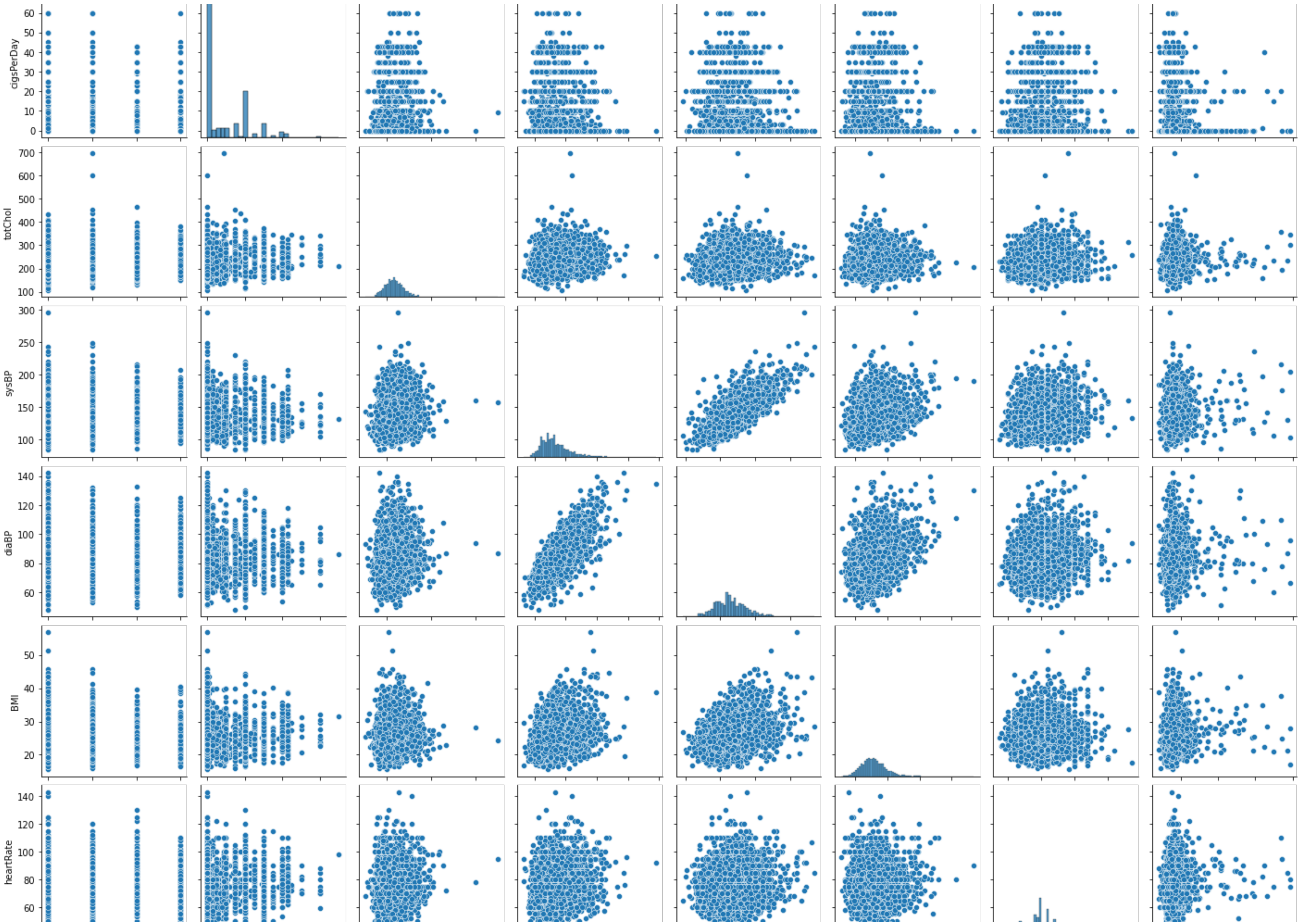
Out[43]:
```
male               False
age                False
education          False
currentSmoker      False
cigsPerDay         False
BPMeds             False
prevalentStroke    False
prevalentHyp       False
diabetes           False
totChol            False
sysBP              False
diaBP              False
BMI                False
heartRate          False
glucose            False
TenYearCHD         False
dtype: bool
```

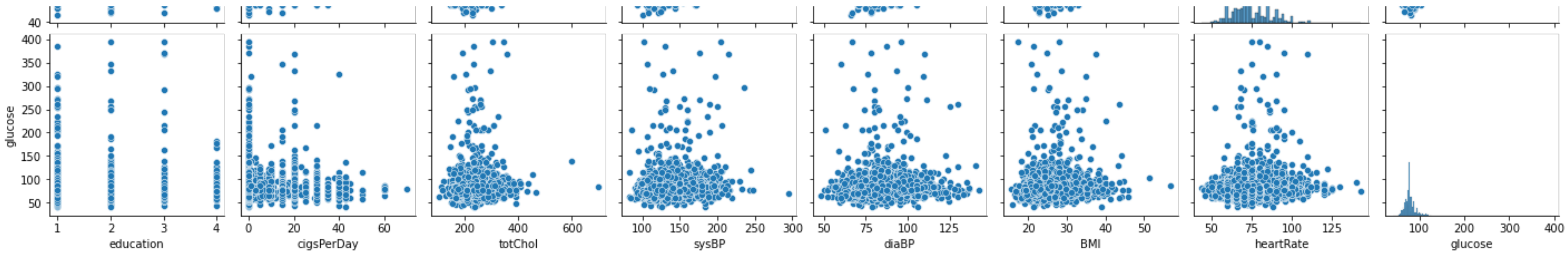
Descriptive Analysis

In [44]:
```
import seaborn as sns
sns.pairplot(df2,vars=['education', 'cigsPerDay', 'totChol','sysBP', 'diaBP', 'BMI', 'heartRate', 'gluco
```

Out[44]:
```
<seaborn.axisgrid.PairGrid at 0x7ff202b26430>
```



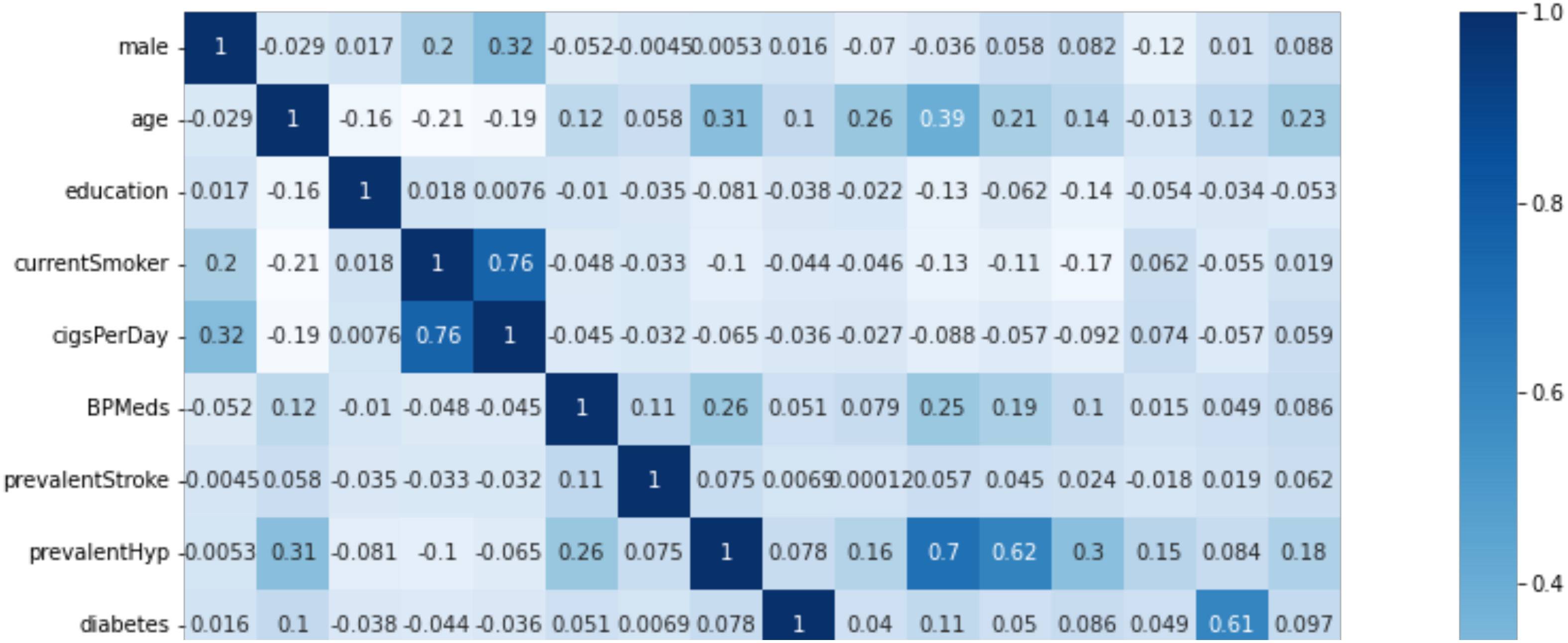

cigsPerDay
totChol
sysBP
diaBP
BMI
heartRate

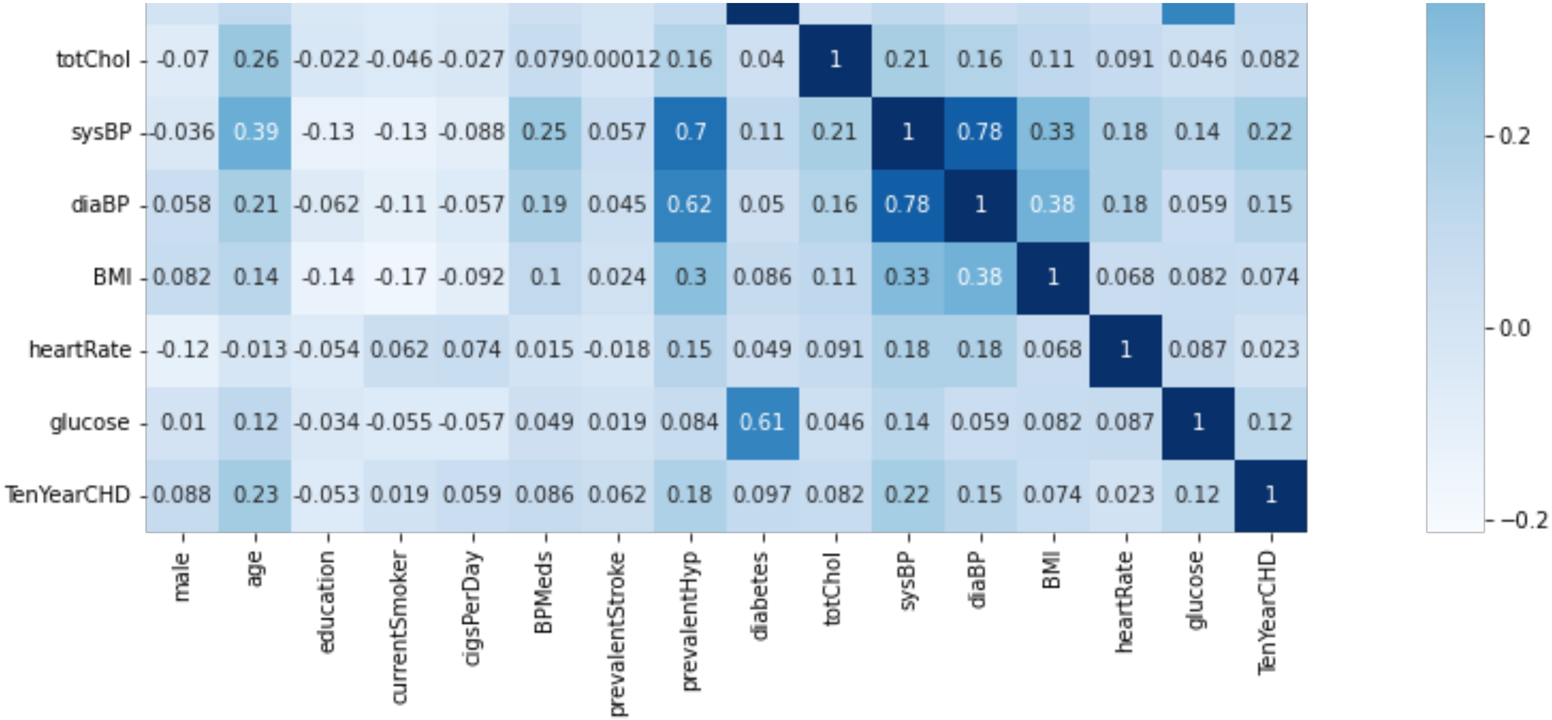


**** Have not plotted for the binary variables ****

In [45]:
```
plt.figure(figsize=(20,10))
c= df2.corr()
sns.heatmap(c,cmap="Blues",annot=True,square = True)
```

Out[45]: <AxesSubplot:>

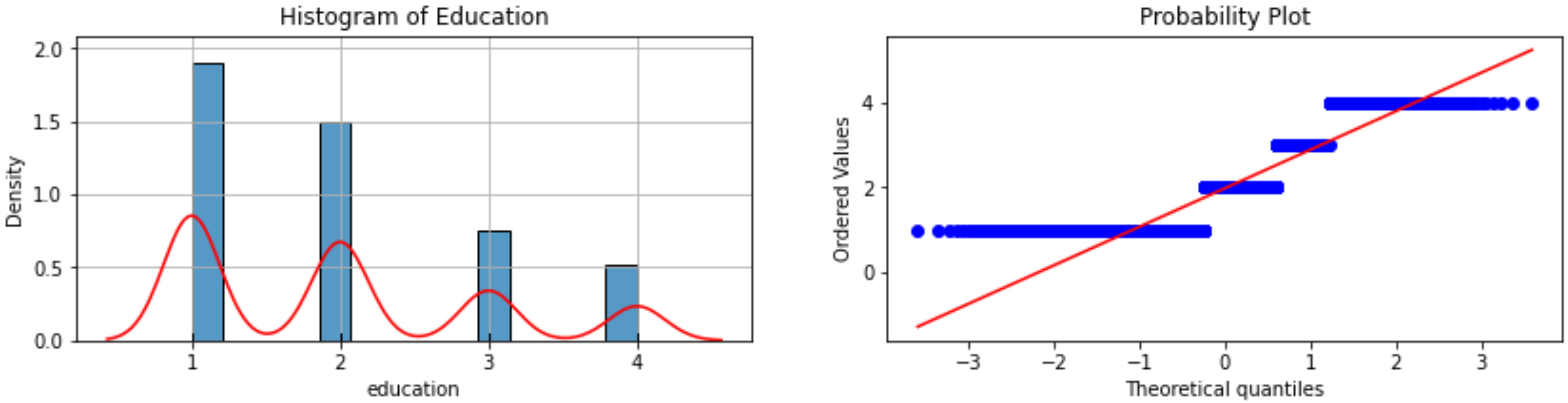




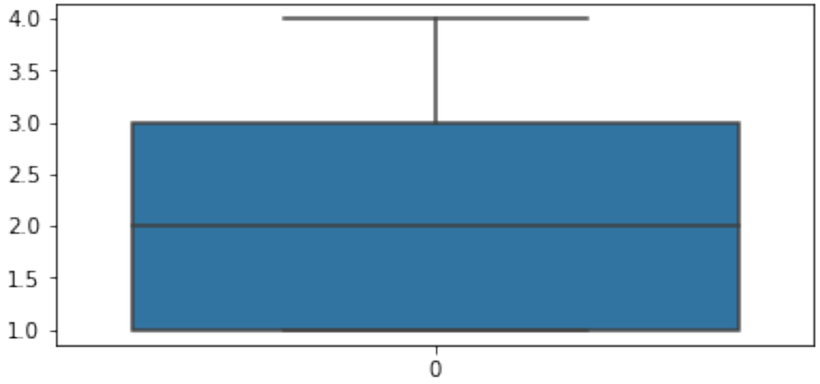
totChol
sysBP
diaBP
BMI
heartRate
glucose
TenYearCHD
-0.07 0.26 -0.022 -0.046 -0.027 0.079 0.00012 0.16 0.04 1 0.21 0.16 0.11 0.091 0.046 0.082
-0.036 0.39 -0.13 -0.13 -0.088 0.25 0.057 0.7 0.11 0.21 1 0.78 0.33 0.18 0.14 0.22
0.058 0.21 -0.062 -0.11 -0.057 0.19 0.045 0.62 0.05 0.16 0.78 1 0.38 0.18 0.059 0.15
0.082 0.14 -0.14 -0.17 -0.092 0.1 0.024 0.3 0.086 0.11 0.33 0.38 1 0.068 0.082 0.074
-0.12 -0.013 -0.054 0.062 0.074 0.015 -0.018 0.15 0.049 0.091 0.18 0.18 0.068 1 0.087 0.023
0.01 0.12 -0.034 -0.055 -0.057 0.049 0.019 0.084 0.61 0.046 0.14 0.059 0.082 0.087 1 0.12
0.088 0.23 -0.053 0.019 0.059 0.086 0.062 0.18 0.097 0.082 0.22 0.15 0.074 0.023 0.12 1
male
age
education
currentSmoker
cigsPerDay
BPMeds
prevalentStroke
prevalentHyp
diabetes
totChol
sysBP
diaBP
BMI
heartRate
glucose
TenYearCHD
0.2
0.0
−0.2

In [46]:

```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Education")
sns.histplot(df2.education, stat = "density")
sns.kdeplot(df2.education, color = "red")
sns.rugplot(df2.education, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.education, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['education'])
```

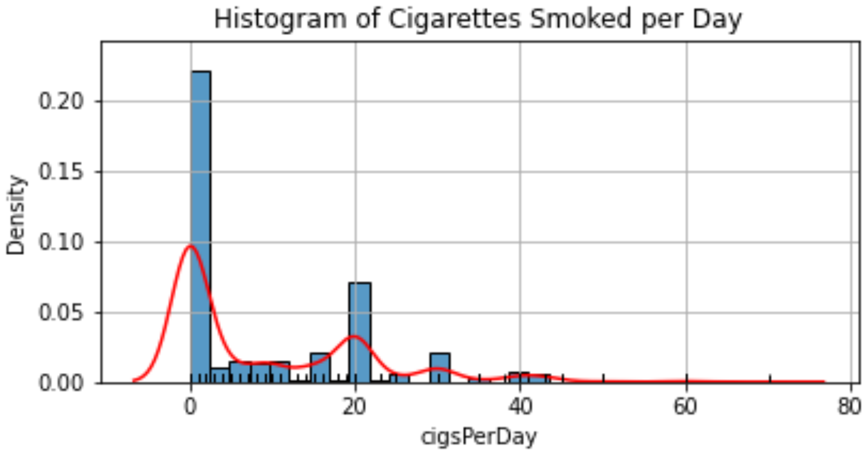


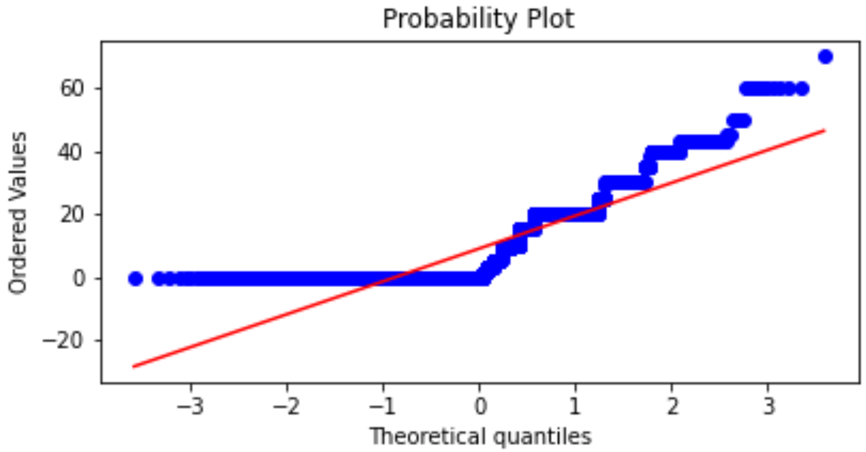
Out[46]: <AxesSubplot:>



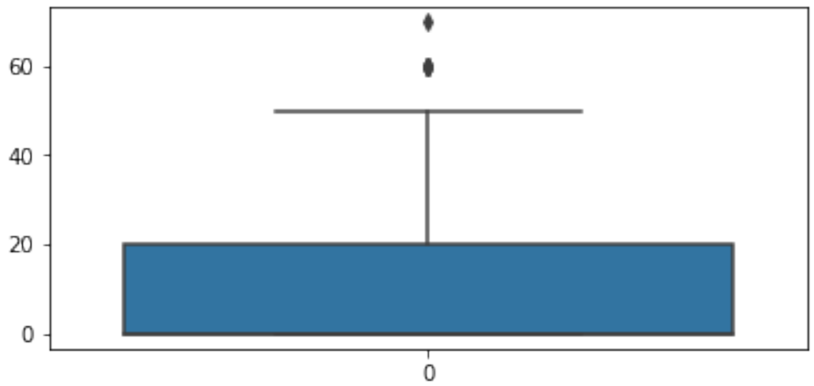
In [47]:

```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Cigarettes Smoked per Day")
sns.histplot(df2.cigsPerDay, stat = "density")
sns.kdeplot(df2.cigsPerDay, color = "red")
sns.rugplot(df2.cigsPerDay, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.cigsPerDay, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['cigsPerDay'])
```



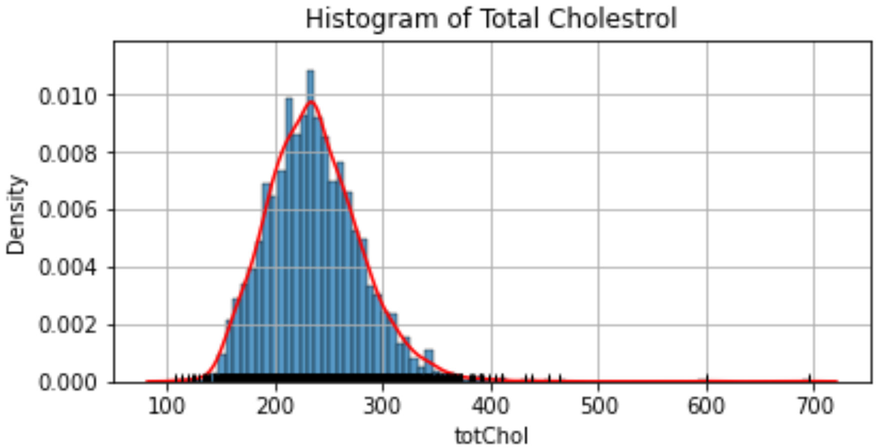


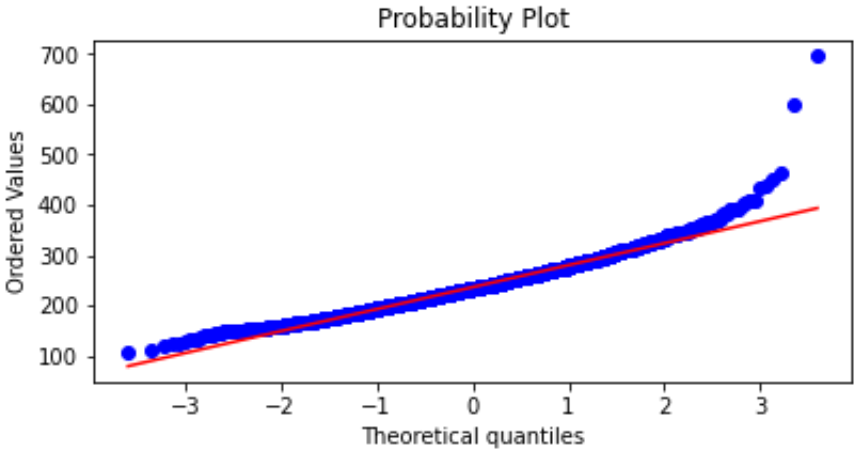
Out[47]: <AxesSubplot:>



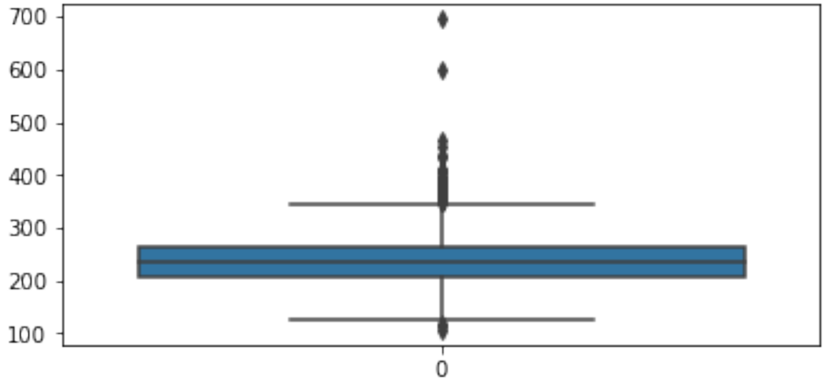
In [48]:

```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Total Cholestrol")
sns.histplot(df2.totChol, stat = "density")
sns.kdeplot(df2.totChol, color = "red")
sns.rugplot(df2.totChol, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.totChol, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['totChol'])
```



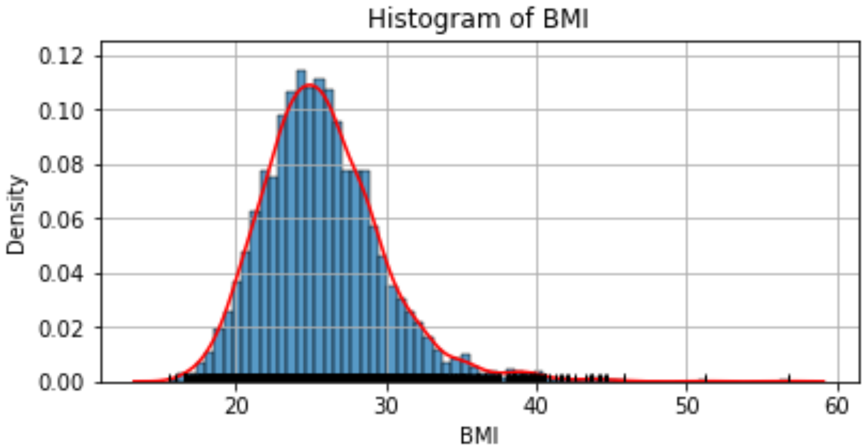


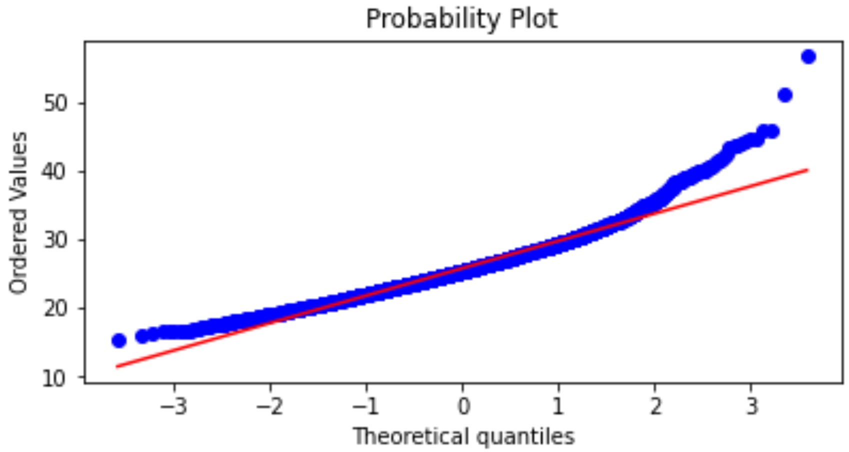
Out[48]: <AxesSubplot:>



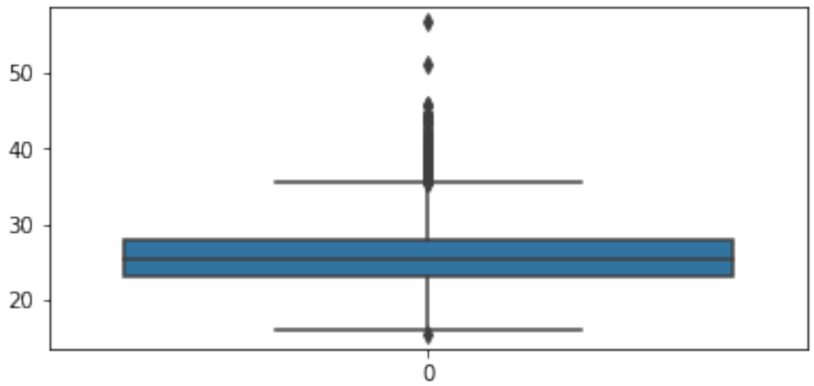
In [49]:

```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of BMI")
sns.histplot(df2.BMI, stat = "density")
sns.kdeplot(df2.BMI, color = "red")
sns.rugplot(df2.BMI, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.BMI, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['BMI'])
```



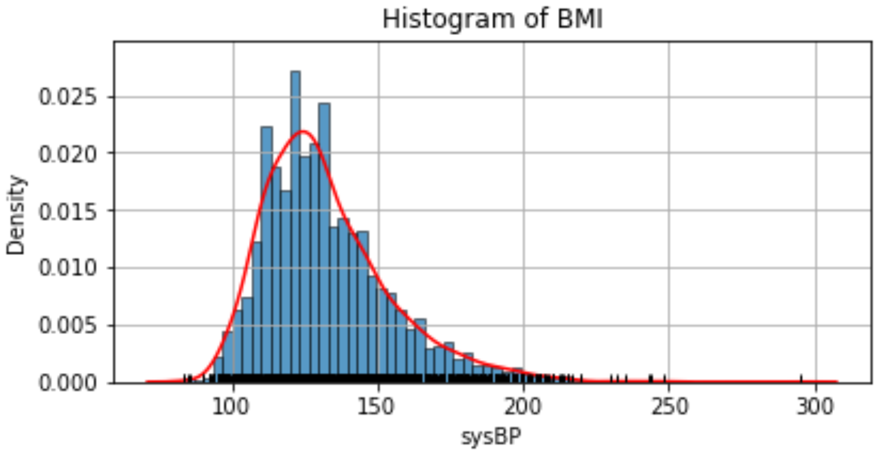


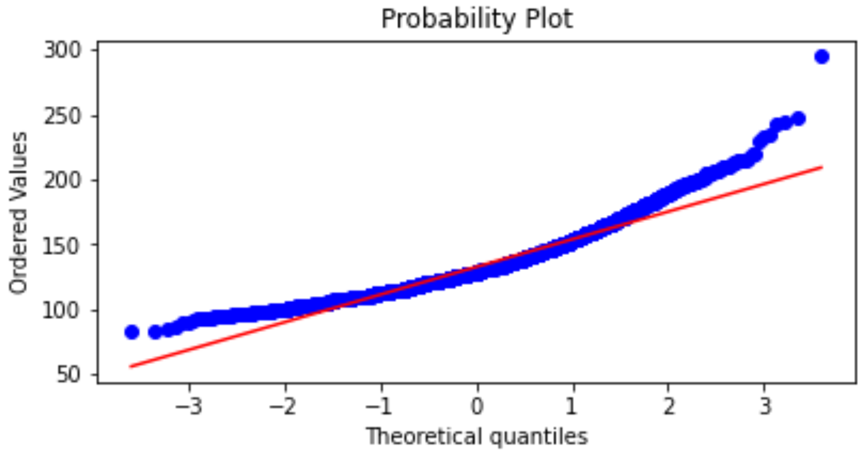
Out[49]: <AxesSubplot:>



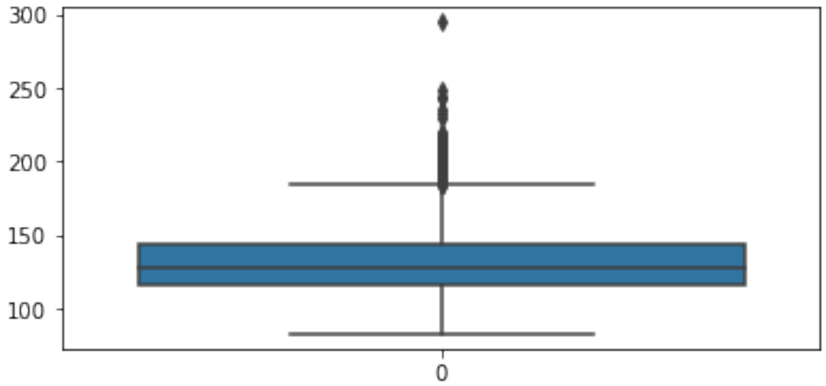
In [50]:

```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of BMI")
sns.histplot(df2.sysBP, stat = "density")
sns.kdeplot(df2.sysBP, color = "red")
sns.rugplot(df2.sysBP, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.sysBP, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['sysBP'])
```



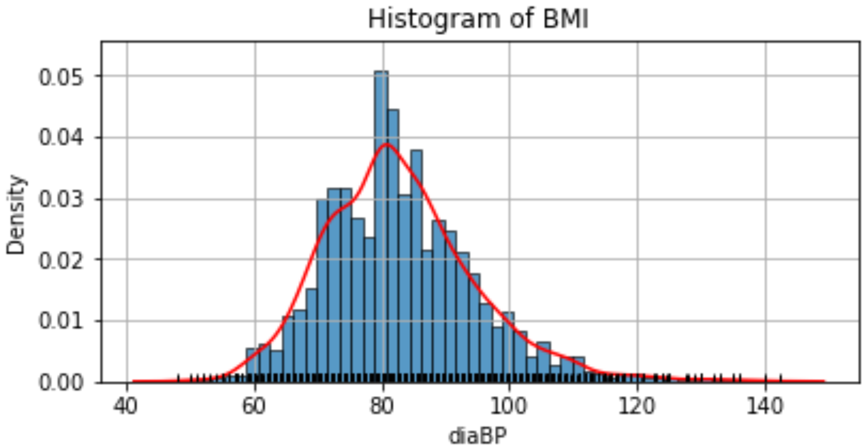


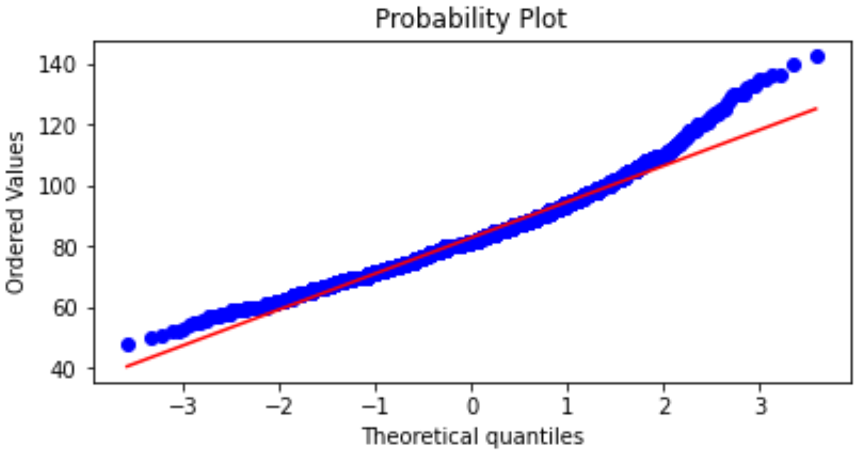
Out[50]: <AxesSubplot:>



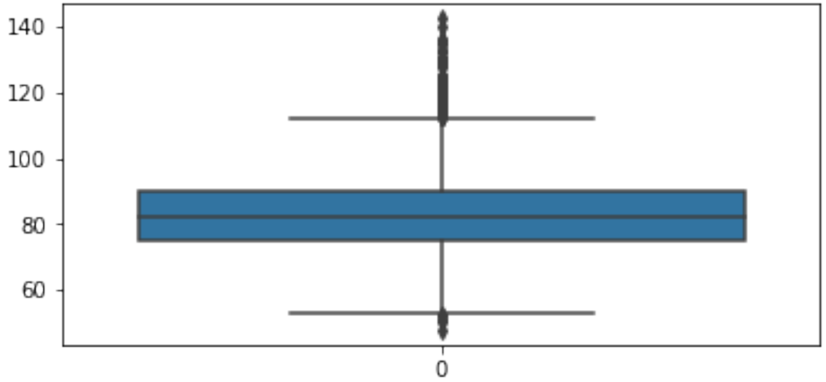
In [51]:

```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of BMI")
sns.histplot(df2.diaBP, stat = "density")
sns.kdeplot(df2.diaBP, color = "red")
sns.rugplot(df2.diaBP, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.diaBP, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['diaBP'])
```



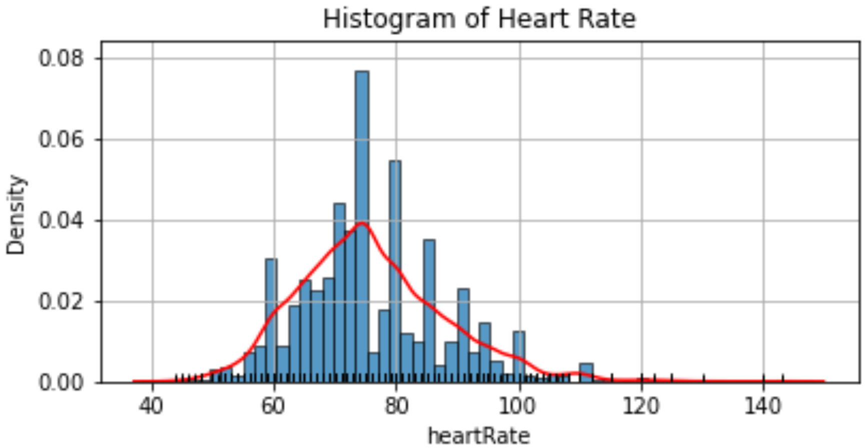


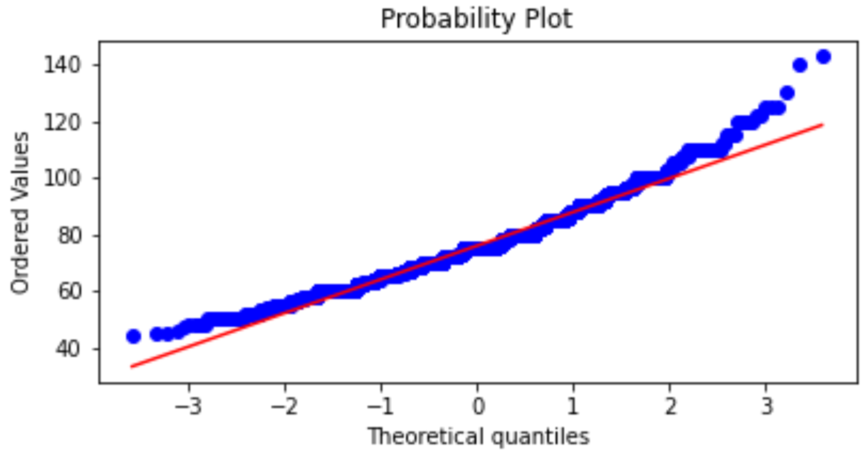
Out[51]: <AxesSubplot:>



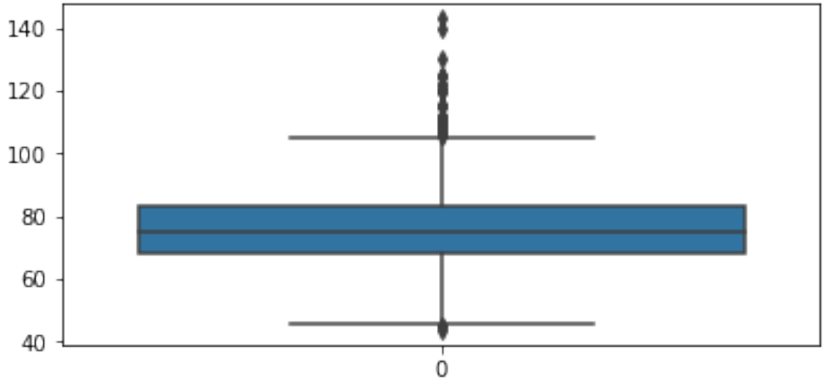
In [52]:

```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Heart Rate")
sns.histplot(df2.heartRate, stat = "density")
sns.kdeplot(df2.heartRate, color = "red")
sns.rugplot(df2.heartRate, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.heartRate, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['heartRate'])
```



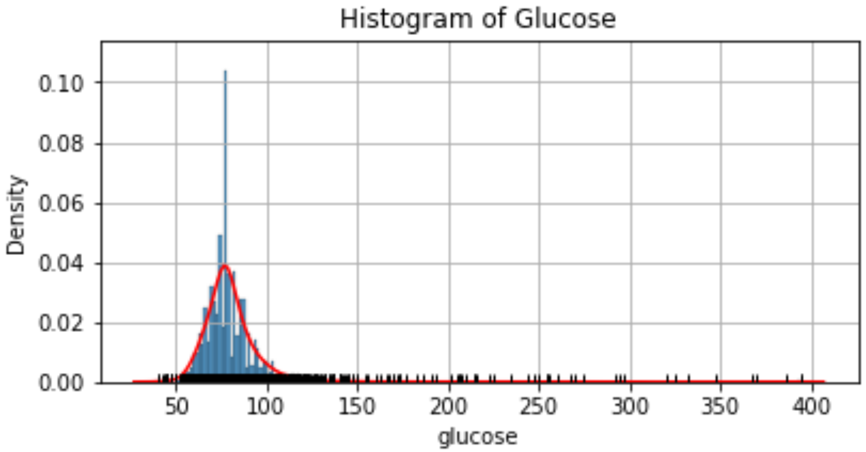


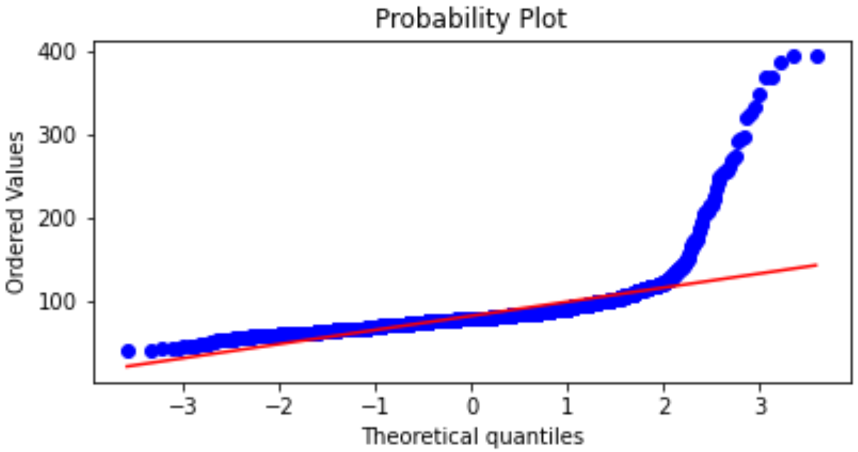
Out[52]: <AxesSubplot:>



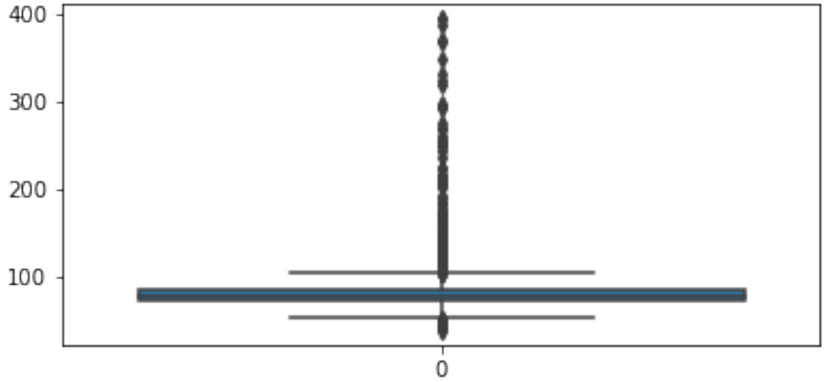
In [53]:

```
plt.figure(figsize = (14,10))

plt.subplot(3,2,1)

plt.title("Histogram of Glucose")
sns.histplot(df2.glucose, stat = "density")
sns.kdeplot(df2.glucose, color = "red")
sns.rugplot(df2.glucose, color = "black")

plt.grid()

plt.subplot(3,2,2)


stats.probplot(df2.glucose, dist="norm", plot=plt)
plt.show()

plt.figure(figsize = (14,10))

plt.subplot(3,2,3)

sns.boxplot(data = df2['glucose'])
```





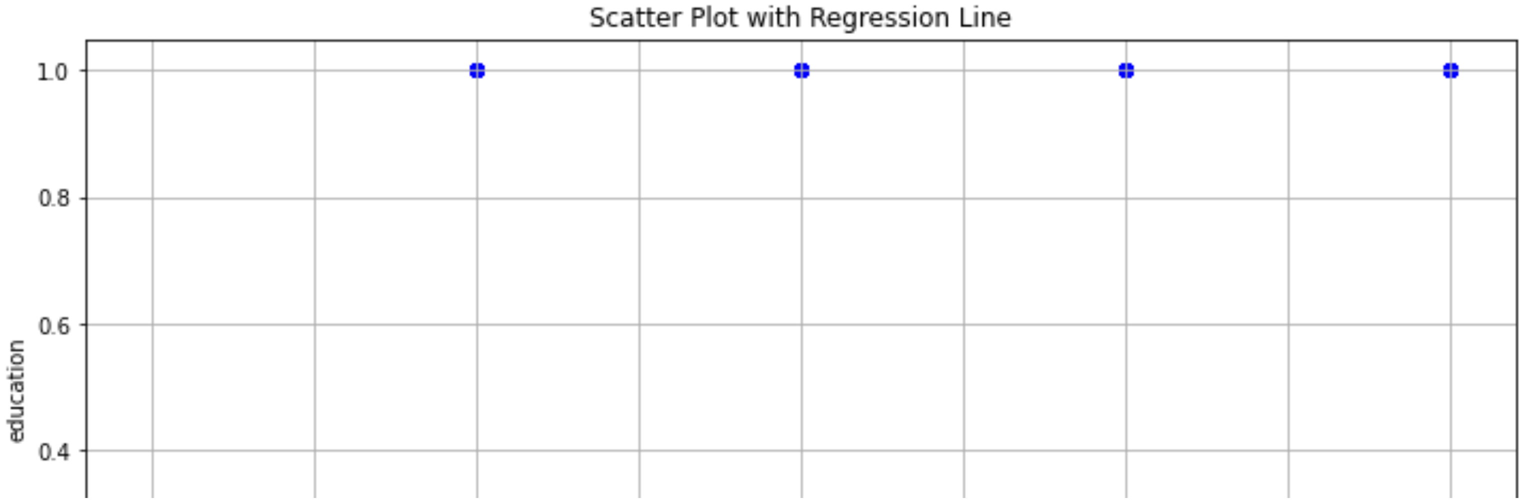
Out[53]: <AxesSubplot:>

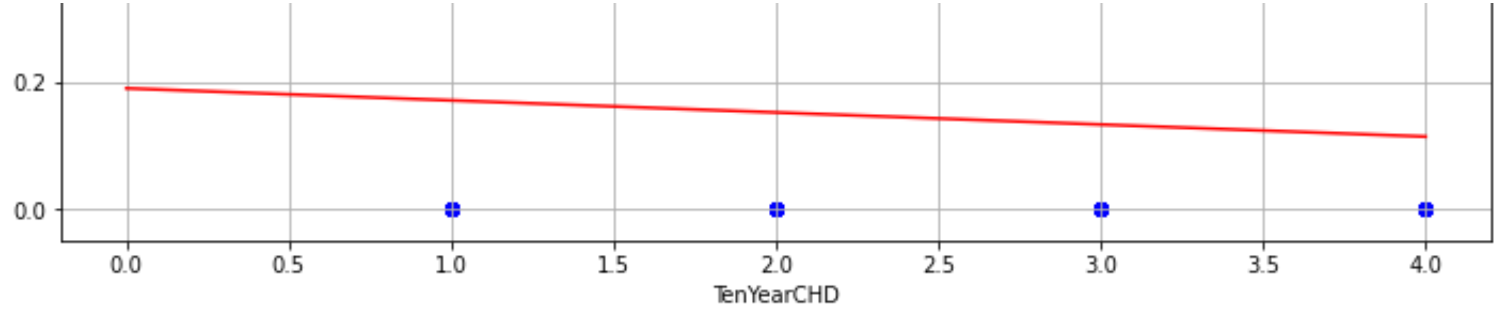


In [65]:

```
plt.figure(figsize = (12, 6))


plt.scatter(df2["education"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["education"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.education.max(), 1000)

# combining the two plots
plt.scatter(df2["education"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("education")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: -0.019030844803086838 The Intercept is: 0.1896294849110646
```

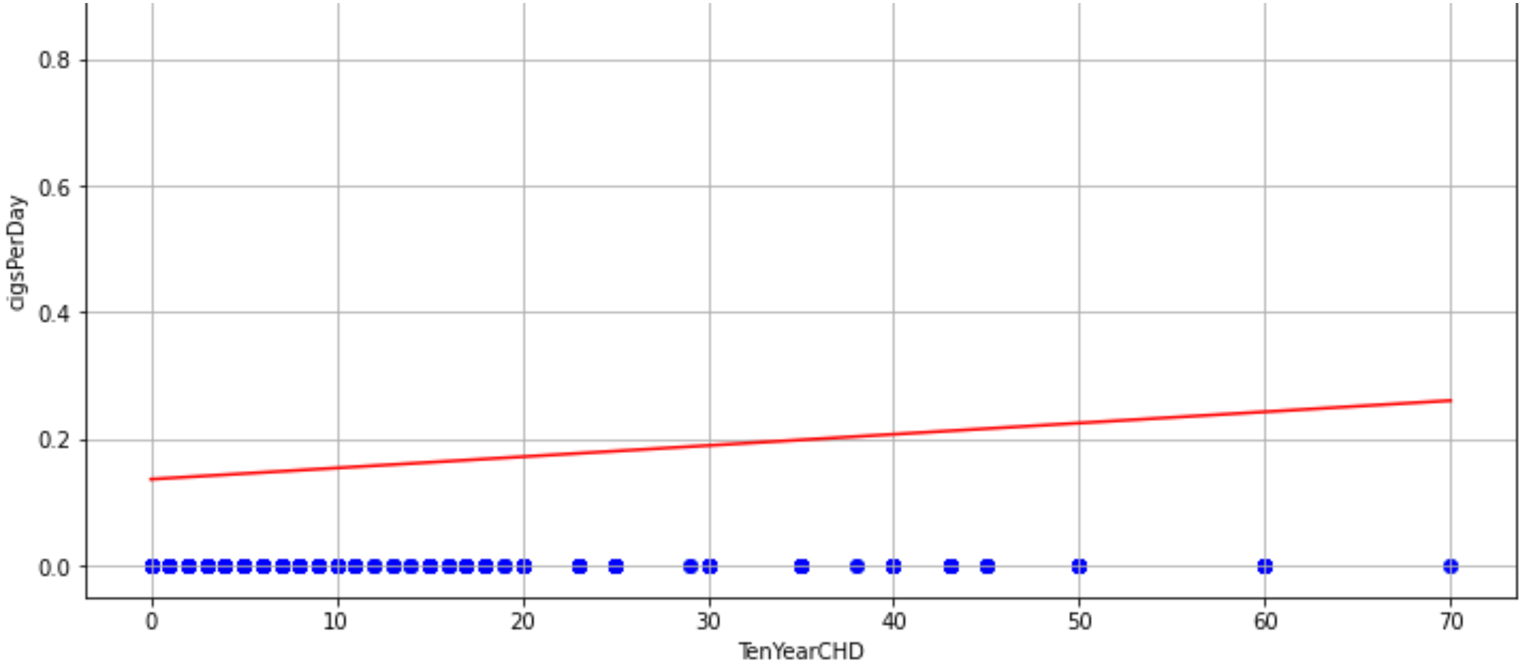




In [67]:

```
plt.figure(figsize = (12, 6))


plt.scatter(df2["cigsPerDay"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["cigsPerDay"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.cigsPerDay.max(), 1000)

# combining the two plots
plt.scatter(df2["cigsPerDay"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("cigsPerDay")
plt.xlabel("TenYearCHD")
plt.grid()
```

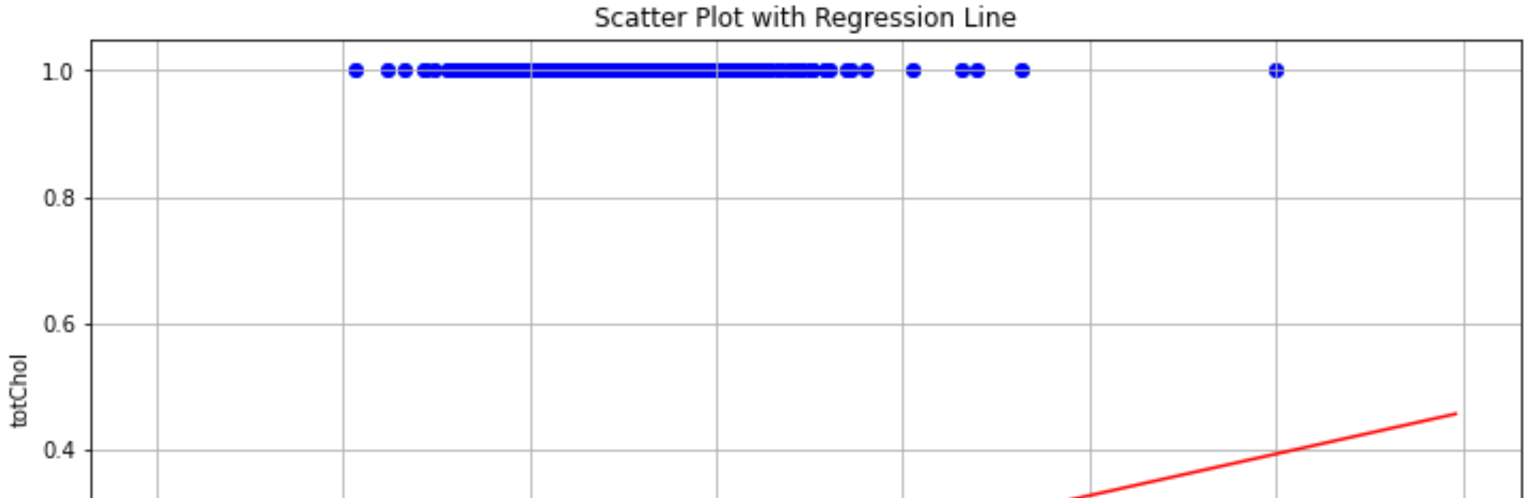
```
The slope of the regression line is: 0.0017754223461071501 The Intercept is: 0.1360835643267134
```

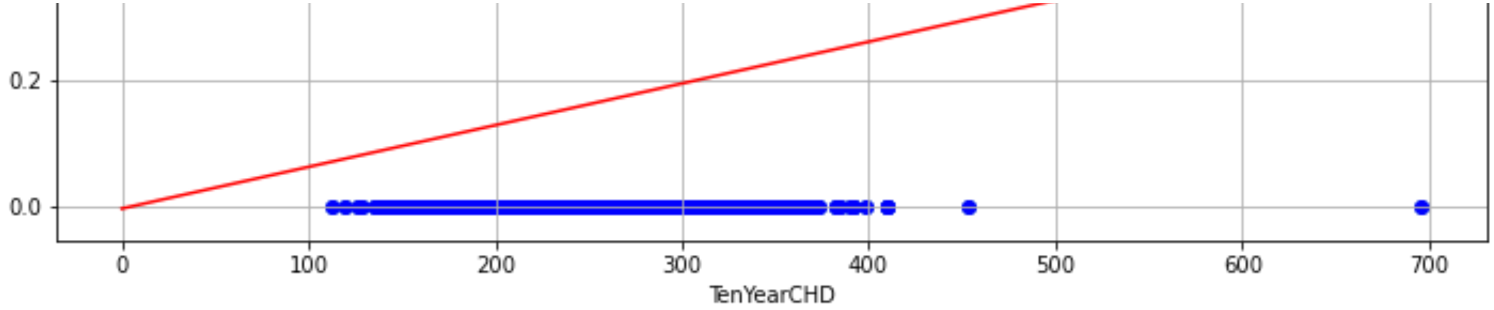


In [122]:

```
plt.figure(figsize = (12, 6))


plt.scatter(df2["totChol"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["totChol"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.totChol.max(), 1000)

# combining the two plots
plt.scatter(df2["totChol"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("totChol")
plt.xlabel("TenYearCHD")
plt.grid()
```

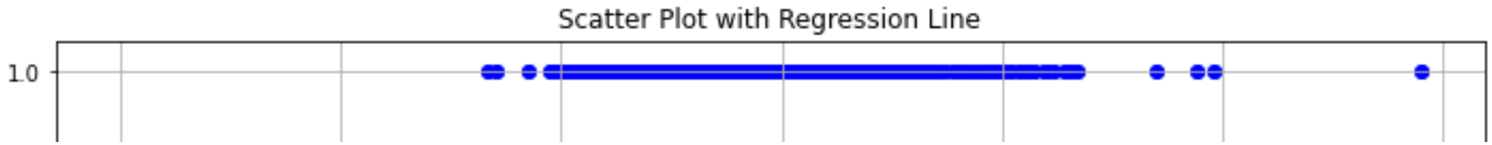
```
The slope of the regression line is: 0.00066062862993396 The Intercept is: -0.004405373383779066
```

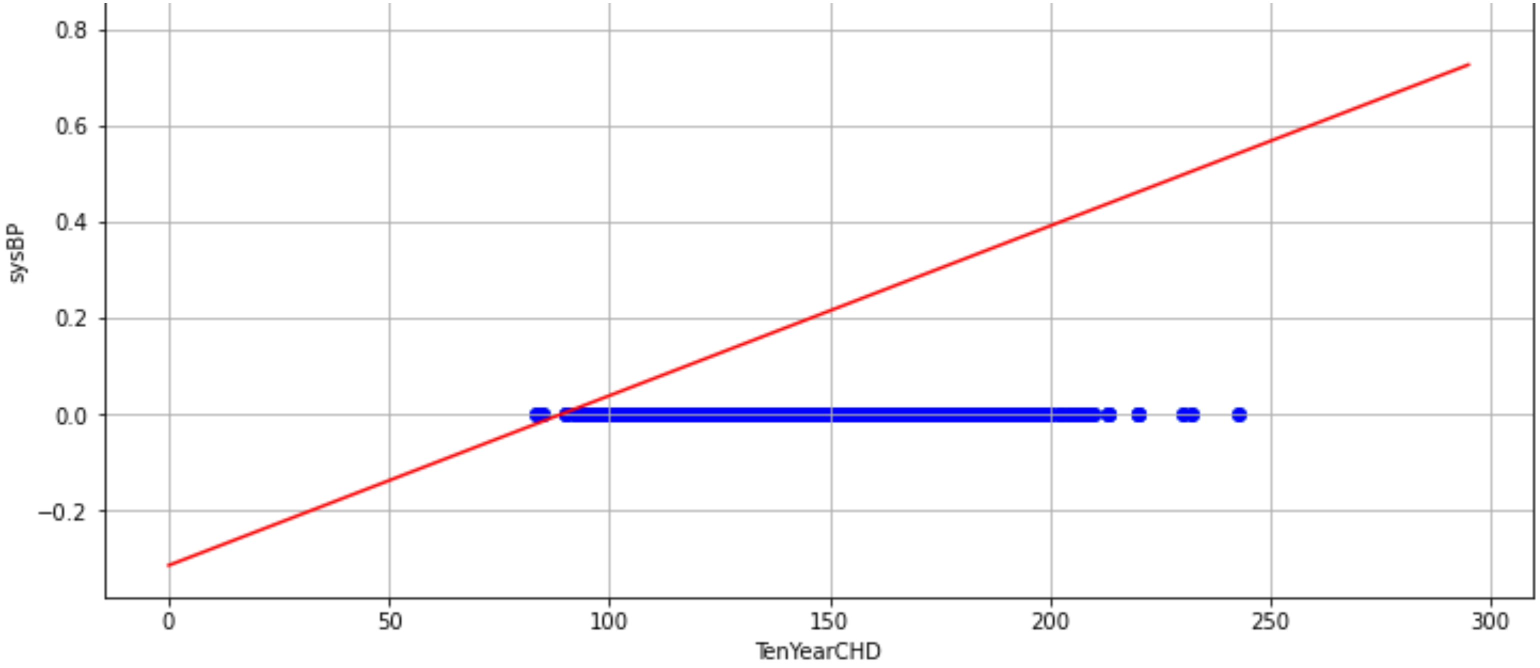




In [69]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["sysBP"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["sysBP"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.sysBP.max(), 1000)

# combining the two plots
plt.scatter(df2["sysBP"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("sysBP")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.0035258490097714794 The Intercept is: -0.3146961314644679
```

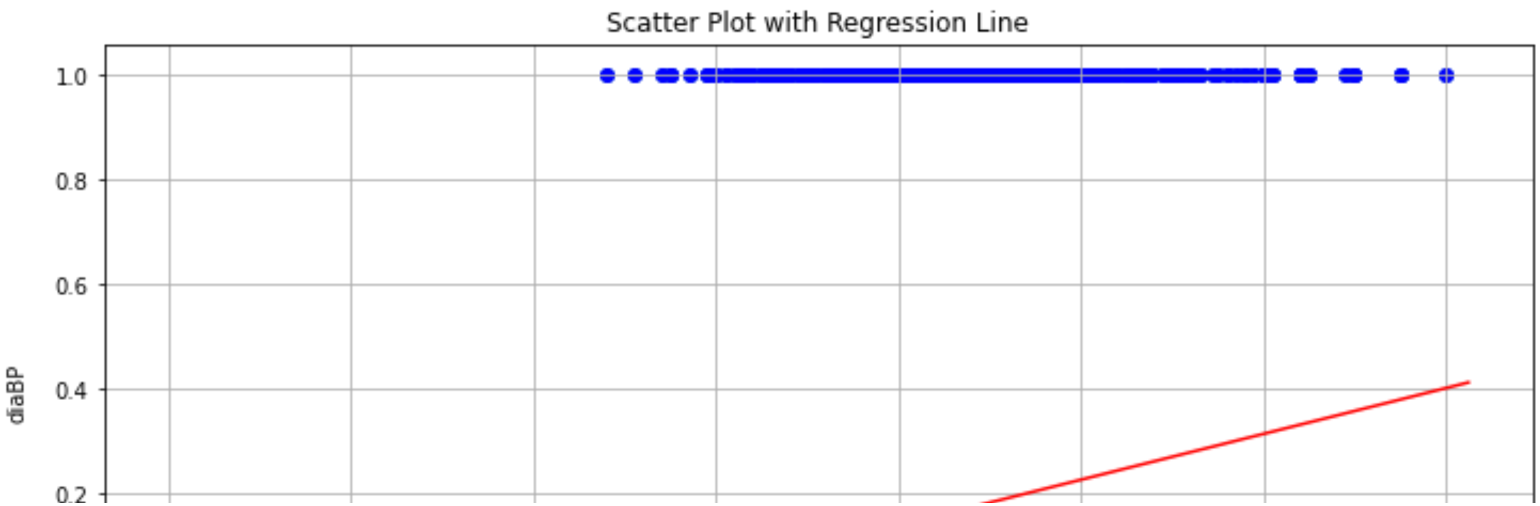


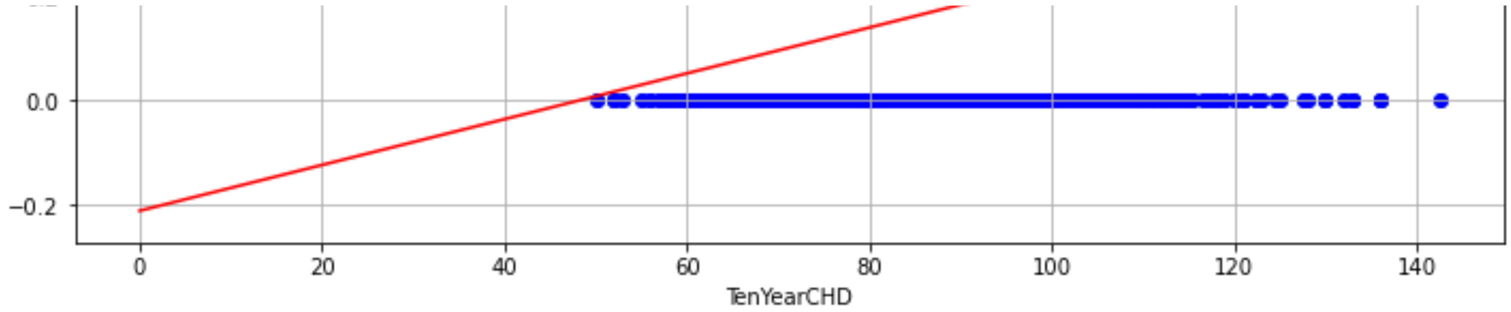


In [70]:

```
plt.figure(figsize = (12, 6))


plt.scatter(df2["diaBP"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["diaBP"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.diaBP.max(), 1000)

# combining the two plots
plt.scatter(df2["diaBP"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("diaBP")
plt.xlabel("TenYearCHD")
plt.grid()
```

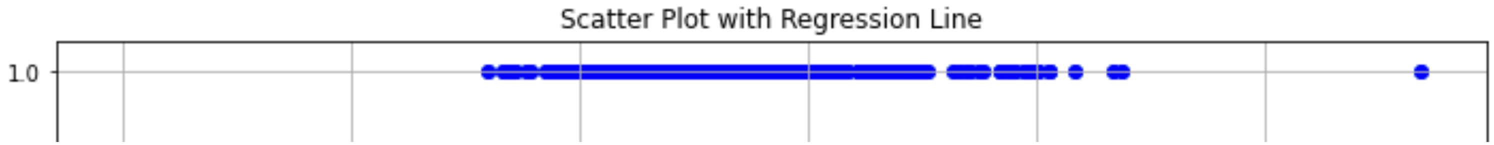
```
The slope of the regression line is: 0.004379680778598945 The Intercept is: -0.21108843952896542
```

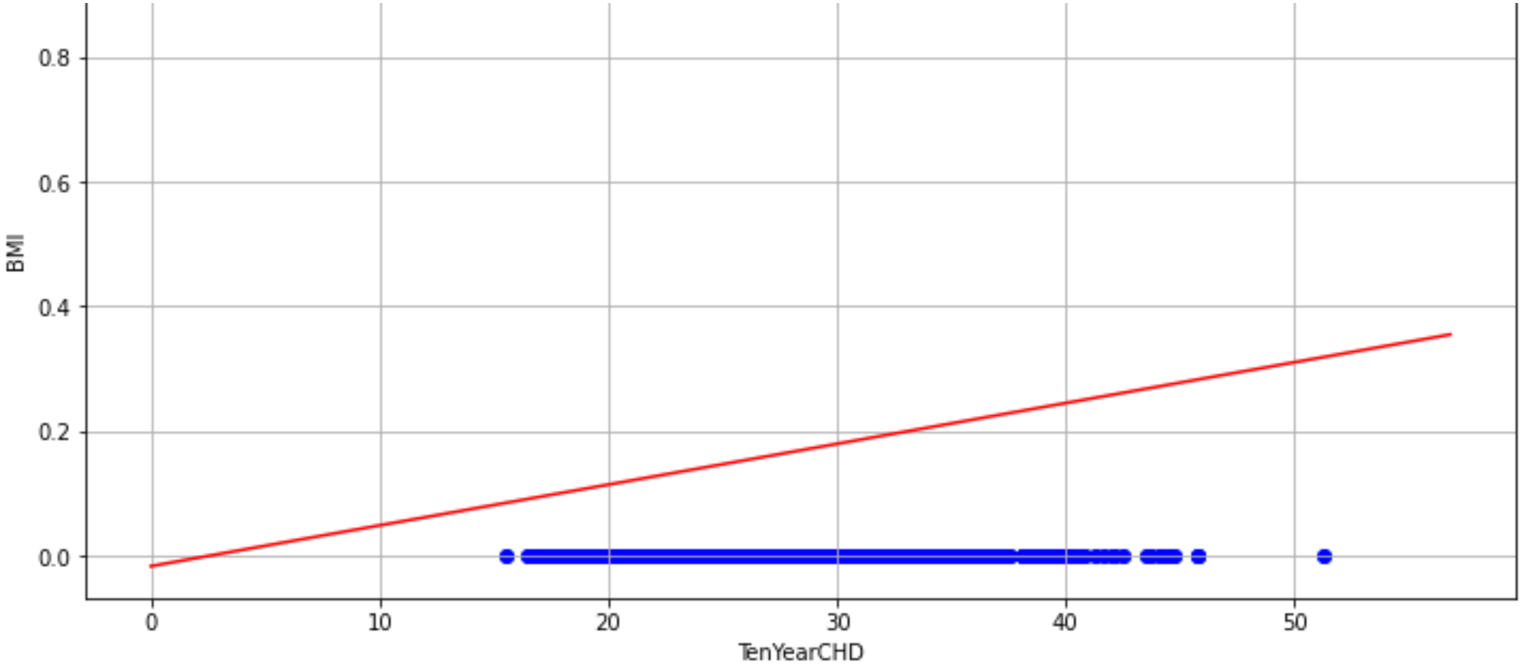




In [71]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["BMI"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["BMI"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.BMI.max(), 1000)

# combining the two plots
plt.scatter(df2["BMI"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("BMI")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.006545124857655006 The Intercept is: -0.016907093969931113
```

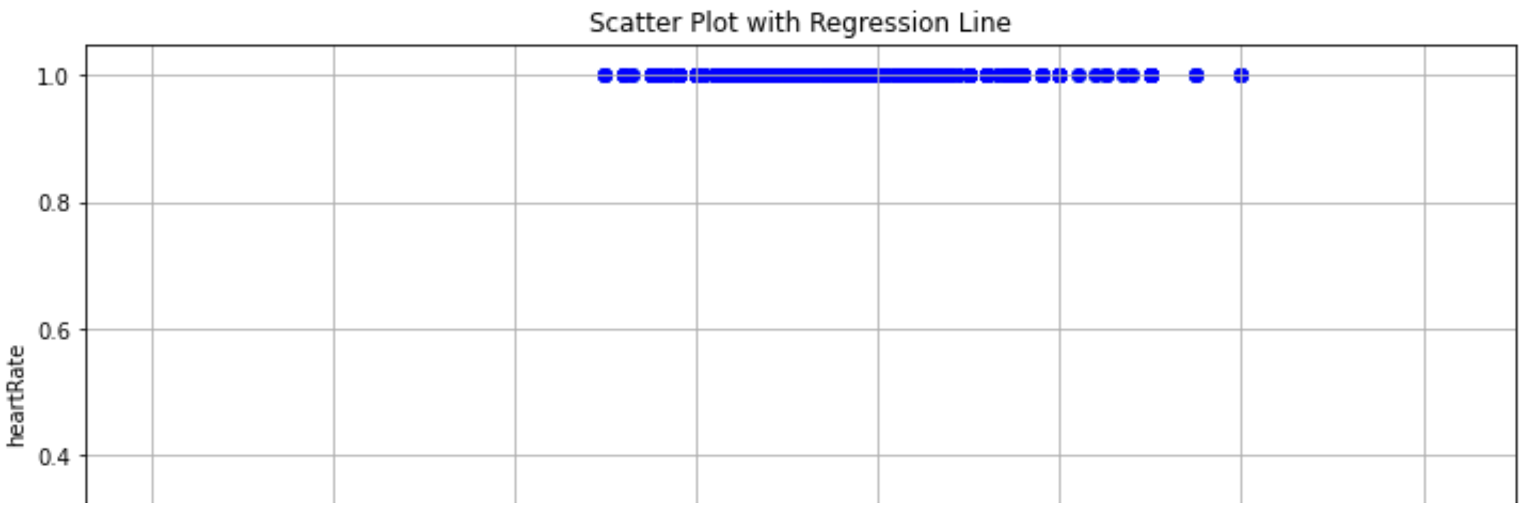


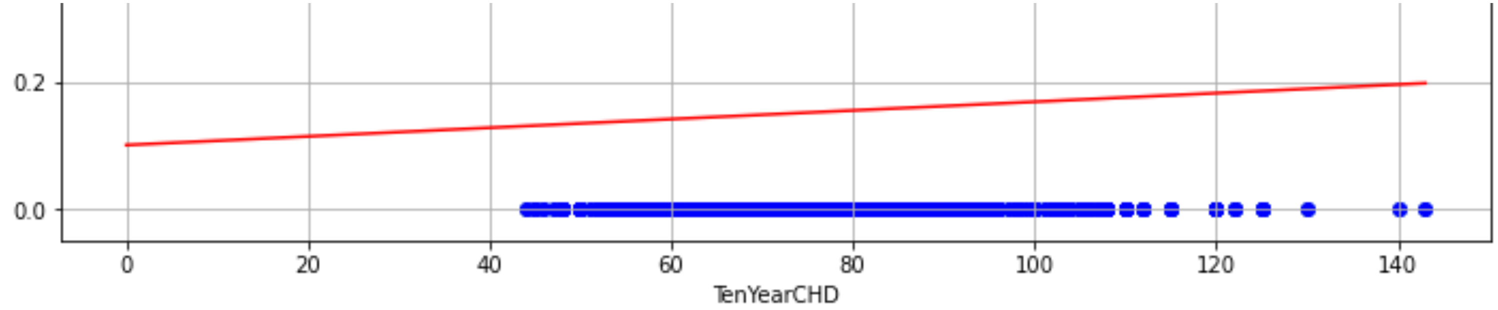


In [72]:

```
plt.figure(figsize = (12, 6))


plt.scatter(df2["heartRate"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["heartRate"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.heartRate.max(), 1000)

# combining the two plots
plt.scatter(df2["heartRate"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("heartRate")
plt.xlabel("TenYearCHD")
plt.grid()
```

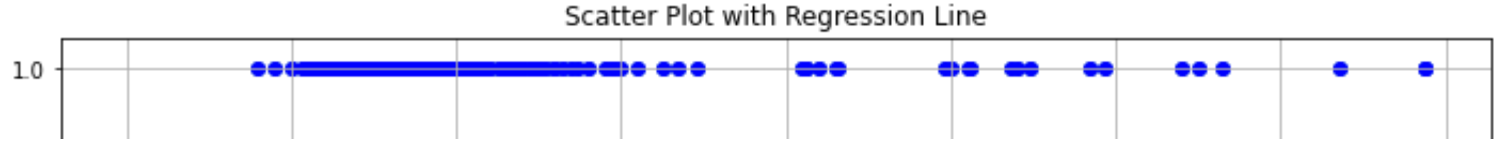
```
The slope of the regression line is: 0.0006824095727596032 The Intercept is: 0.10017810855342241
```

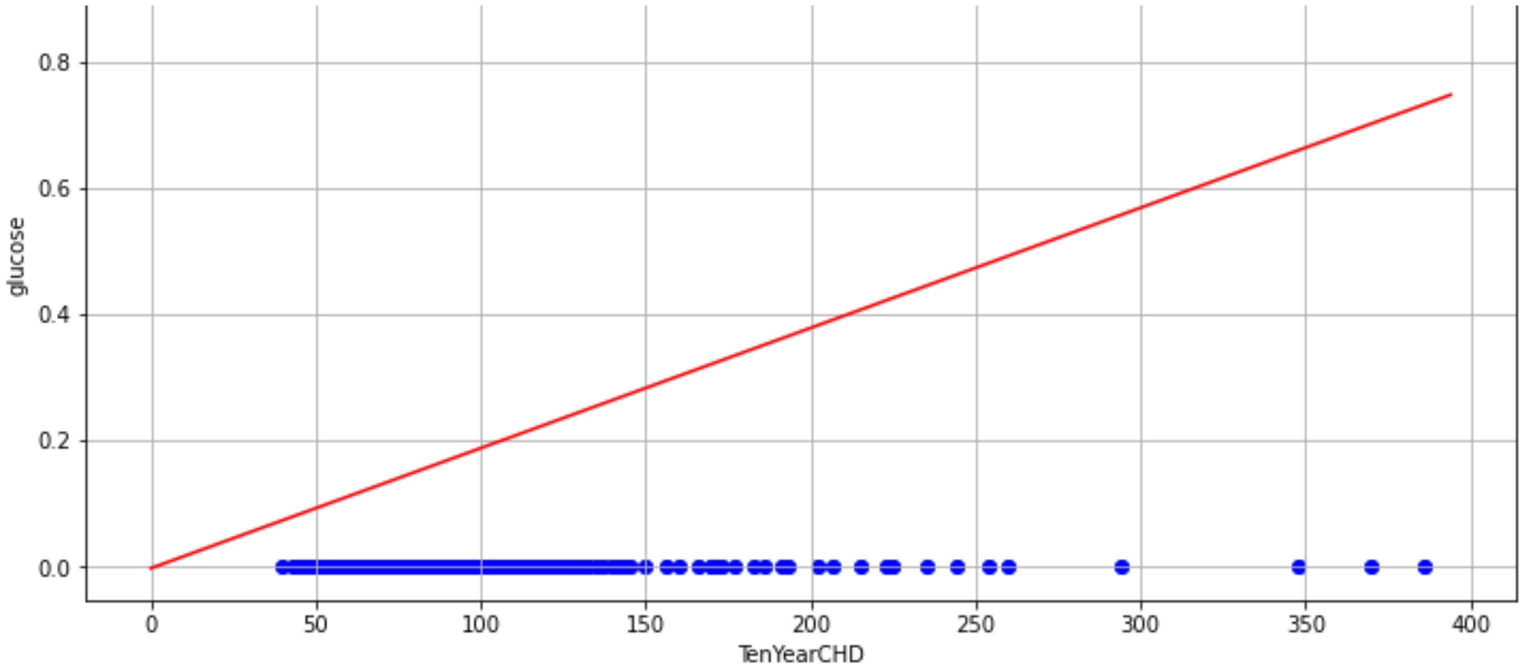




In [73]:
```
plt.figure(figsize = (12, 6))


plt.scatter(df2["glucose"], df2["TenYearCHD"])

# Create regression line
m,b = np.polyfit(df2["glucose"], df2["TenYearCHD"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df2.glucose.max(), 1000)

# combining the two plots
plt.scatter(df2["glucose"], df2["TenYearCHD"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("glucose")
plt.xlabel("TenYearCHD")
plt.grid()
```

```
The slope of the regression line is: 0.0019042600407882346 The Intercept is: -0.003435978165653731
```





In [76]:
```
model = smf.ols(formula='TenYearCHD ~ male + age + education + currentSmoker + cigsPerDay + BPMeds + pre

result1 = model.fit()

print(result1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             TenYearCHD   R-squared:                       0.097
Model:                            OLS   Adj. R-squared:                  0.094
Method:                 Least Squares   F-statistic:                     30.31
Date:                Wed, 30 Nov 2022   Prob (F-statistic):           9.97e-83
Time:                        16:14:40   Log-Likelihood:                -1454.9
No. Observations:                4238   AIC:                             2942.
Df Residuals:                    4222   BIC:                             3044.
Df Model:                          15
```

```
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept         -0.5592      0.074     -7.557      0.000      -0.704      -0.414
male               0.0529      0.012      4.562      0.000       0.030       0.076
age                0.0069      0.001      9.642      0.000       0.006       0.008
education         -0.0018      0.005     -0.329      0.742      -0.012       0.009
currentSmoker     -0.0005      0.016     -0.028      0.977      -0.033       0.032
cigsPerDay         0.0027      0.001      3.783      0.000       0.001       0.004
BPMeds             0.0550      0.033      1.682      0.093      -0.009       0.119
prevalentStroke    0.1947      0.069      2.814      0.005       0.059       0.330
prevalentHyp       0.0288      0.016      1.770      0.077      -0.003       0.061
diabetes           0.0472      0.042      1.128      0.259      -0.035       0.129
totChol         8.908e-05      0.000      0.715      0.475      -0.000       0.000
sysBP              0.0023      0.000      4.926      0.000       0.001       0.003
diaBP             -0.0010      0.001     -1.270      0.204      -0.002       0.001
BMI               -0.0003      0.001     -0.194      0.846      -0.003       0.003
heartRate         -0.0002      0.000     -0.395      0.693      -0.001       0.001
glucose            0.0011      0.000      3.853      0.000       0.001       0.002
==============================================================================
Omnibus:                     1179.378   Durbin-Watson:                   2.036
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2451.322
Skew:                           1.690   Prob(JB):                         0.00
Kurtosis:                       4.568   Cond. No.                     4.47e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.47e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [77]:
```
import statsmodels.stats.outliers_influence as smo
import patsy as pt

# extract matrices using patsy:
y, X = pt.dmatrices('TenYearCHD ~ male + age + education + currentSmoker + cigsPerDay + BPMeds + prevale

# get VIF:
K = X.shape[1]
VIF = np.empty(K)
for i in range(K):
    VIF[i] = smo.variance_inflation_factor(X.values, i)
print(f'VIF: \n{VIF}\n')
```

```
VIF:
[198.69094864   1.19355883   1.37233419   1.05465334   2.45409906
   2.57426122   1.10095353   1.019168     2.05314668   1.58930635
   1.10697227   3.73870446   2.96534322   1.23552405   1.09571782
   1.61146399]
```

No multicollinearity as all values are below the threshold of 4.

In [78]:
```
import statsmodels.regression.linear_model as rg
import statsmodels.stats.diagnostic as dg

test =  dg.linear_reset(result1, power=2,  test_type='fitted', use_f = True)
test
```

```
/Users/snehilshandilya/opt/anaconda3/lib/python3.9/site-packages/statsmodels/stats/diagnostic.py:1081:
FutureWarning: Support for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be r
emoved in a future version.  Convert to a numpy array before indexing instead.
  aug = res.fittedvalues[:, None]
```

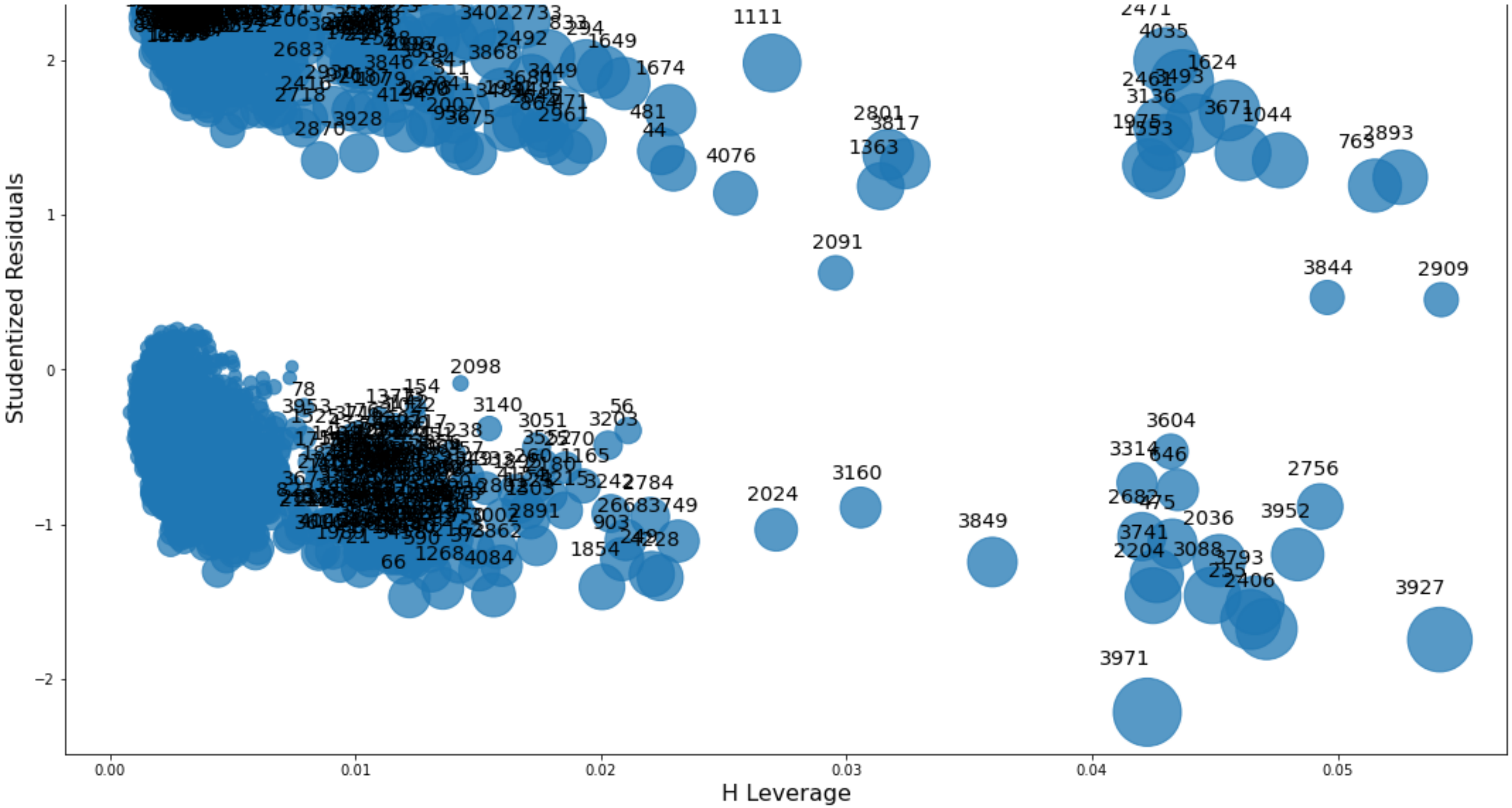
Out[78]:
```
<class 'statsmodels.stats.contrast.ContrastResults'>
<F test: F=33.36273654238361, p=8.197680069437e-09, df_denom=4.22e+03, df_num=1>
```

p-value is below 0.05 ****

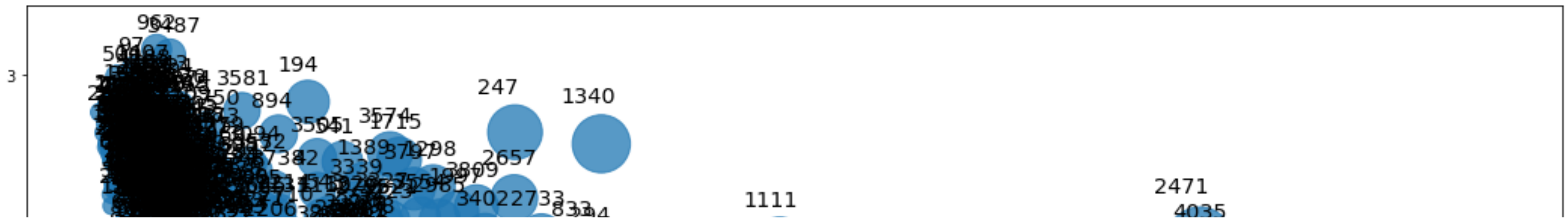
Interpret

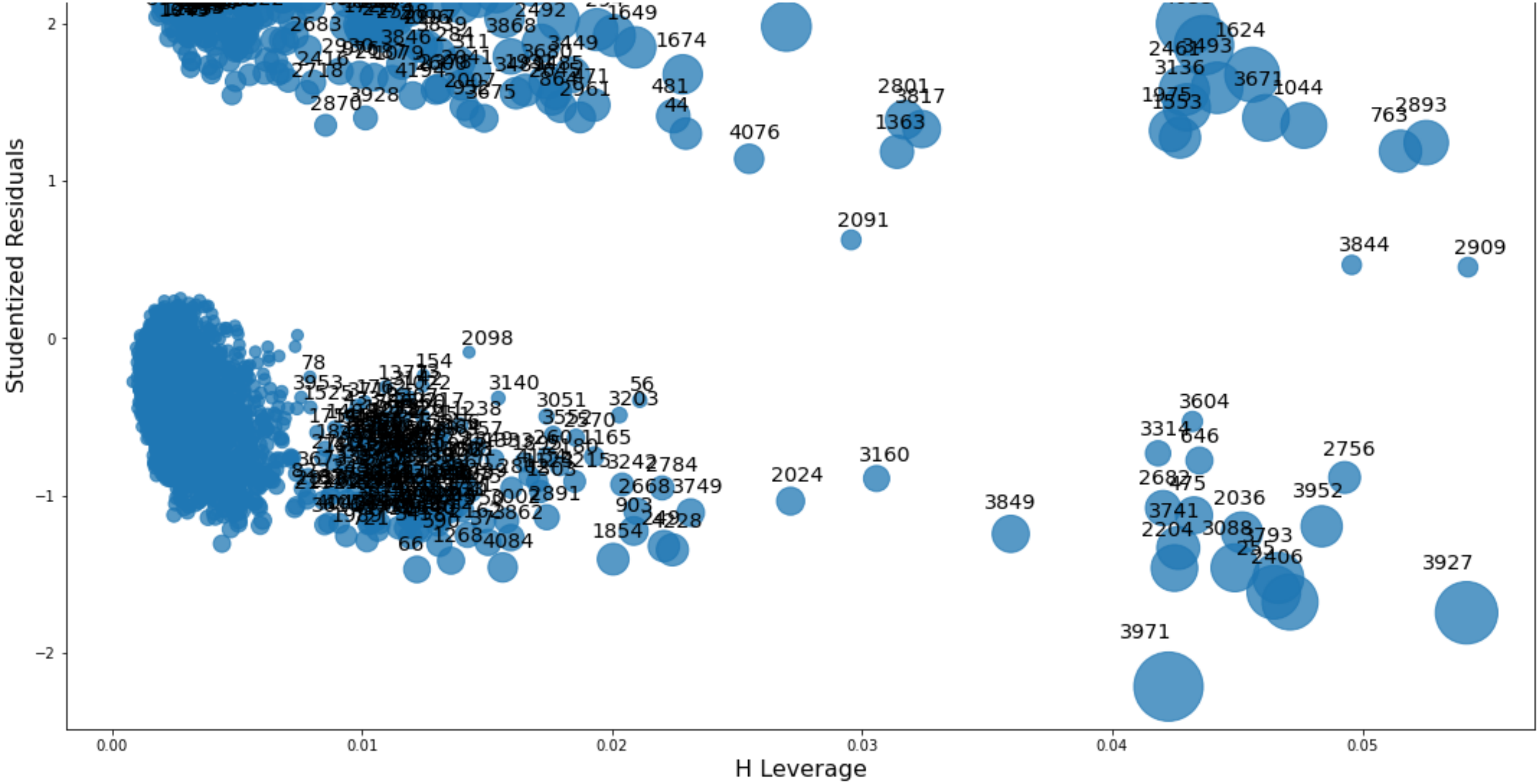
In [80]:
```
figd, ax = plt.subplots(figsize=(15,10))
figd = sm.graphics.influence_plot(result1, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(15,10))
fige = sm.graphics.influence_plot(result1, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```

Influence Plot



Influence Plot



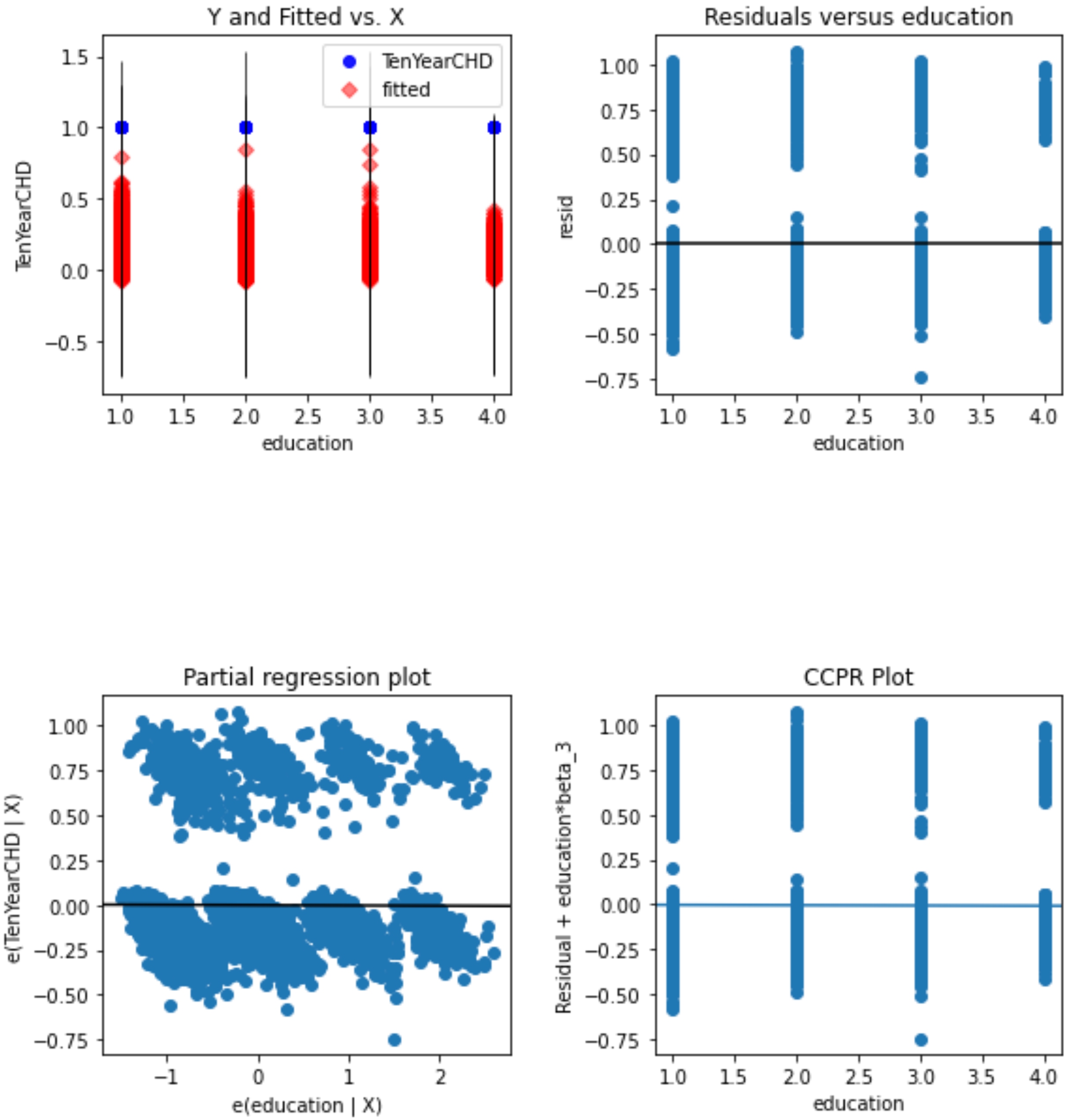


In [81]:

```
fig = sm.graphics.plot_regress_exog(result1, "education")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

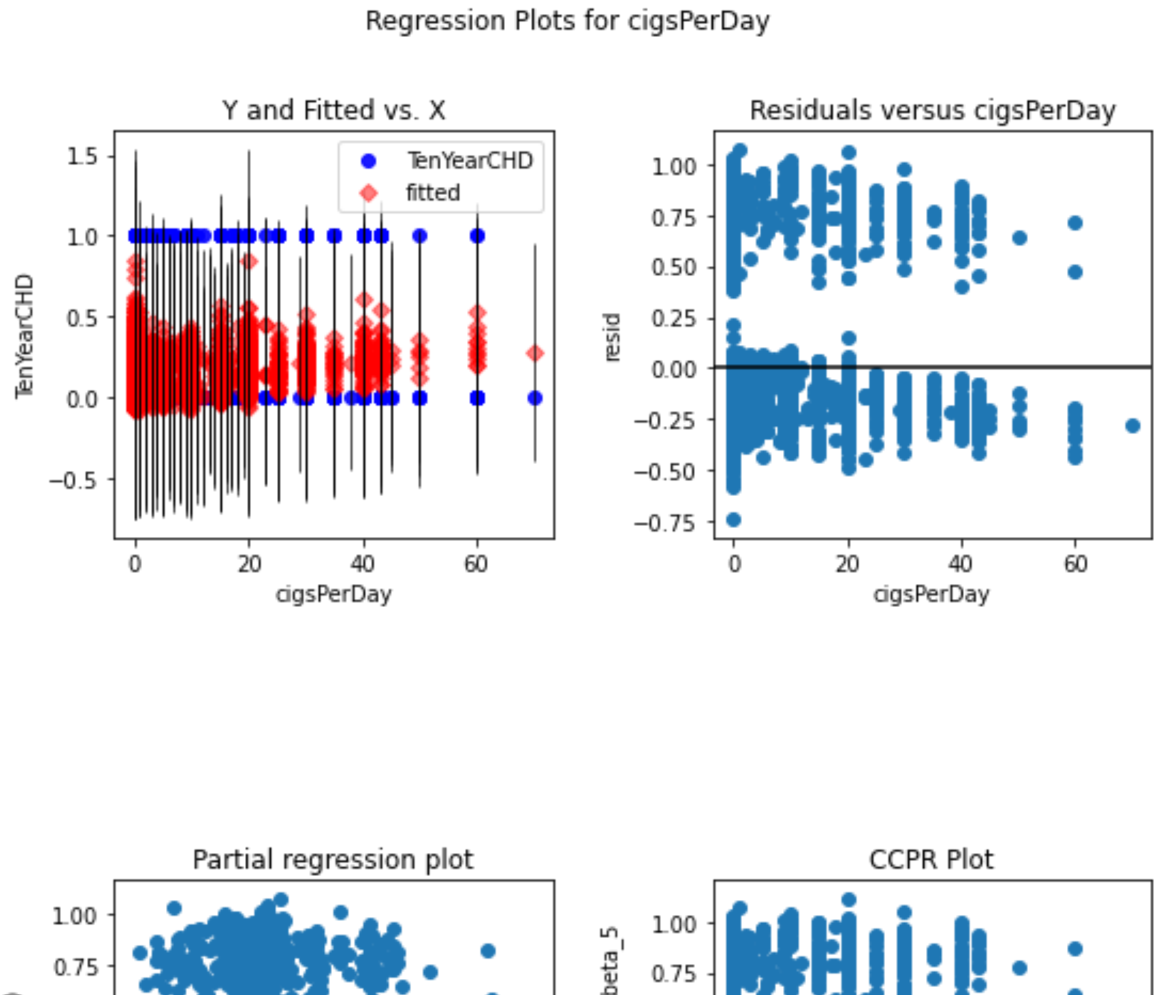
```
eval_env: 1
```

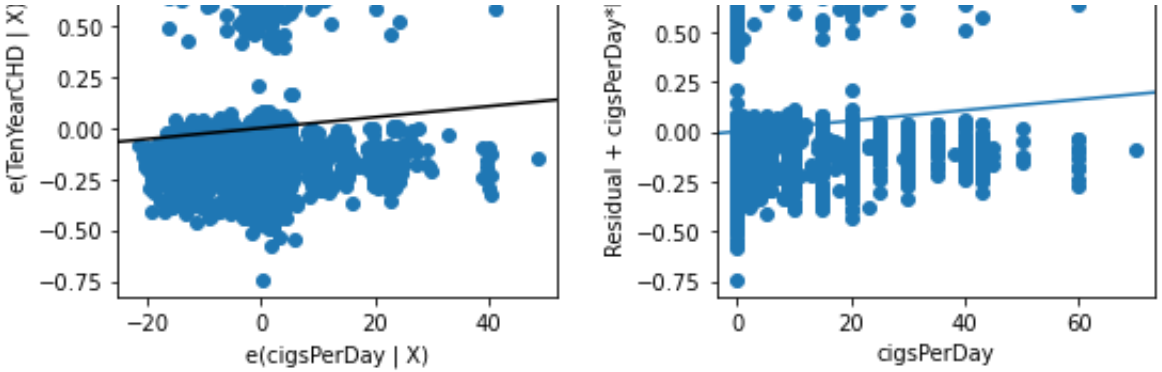
Regression Plots for education


Y and Fitted vs. X
TenYearCHD
fitted
TenYearCHD
education
Residuals versus education
resid
education
Partial regression plot
e(TenYearCHD | X)
e(education | X)
CCPR Plot
Residual + education*beta_3
education

In [82]:
```
fig = sm.graphics.plot_regress_exog(result1, "cigsPerDay")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

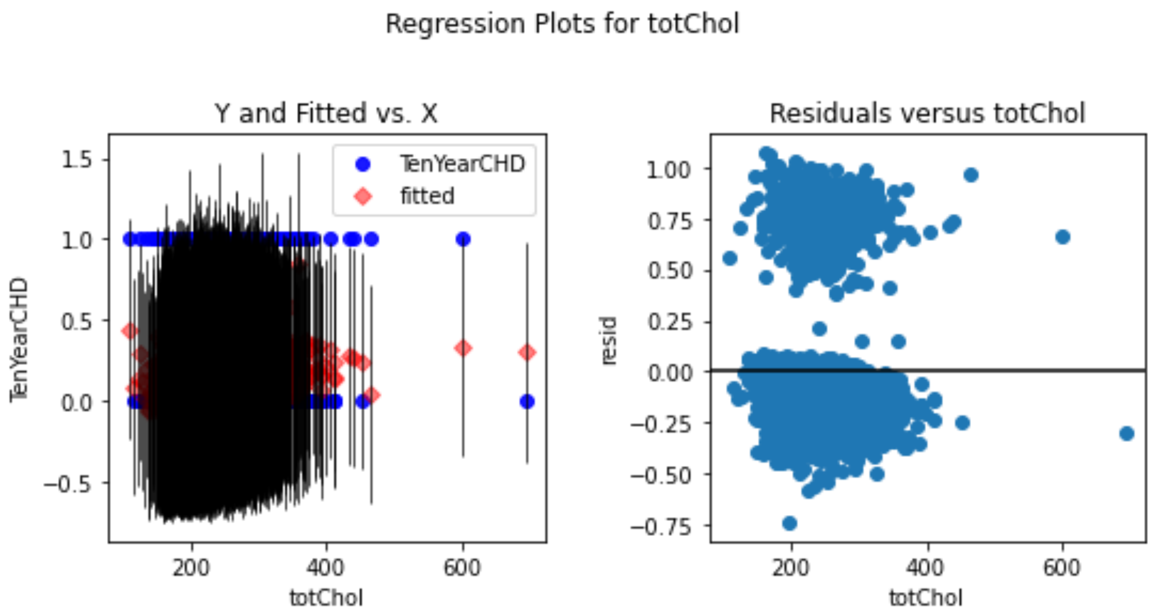
```
eval_env: 1
```

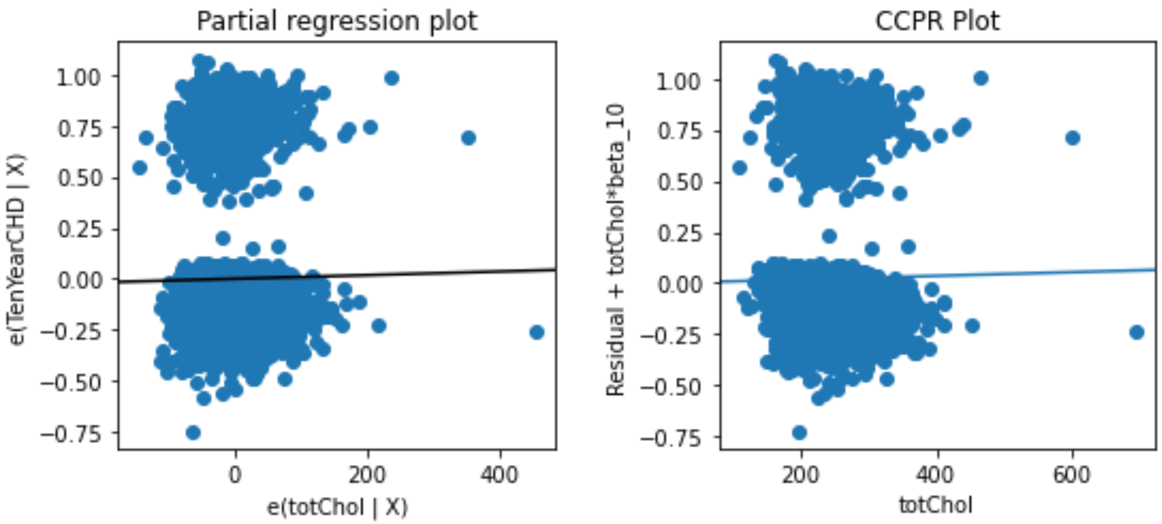




In [83]:
```
fig = sm.graphics.plot_regress_exog(result1, "totChol")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

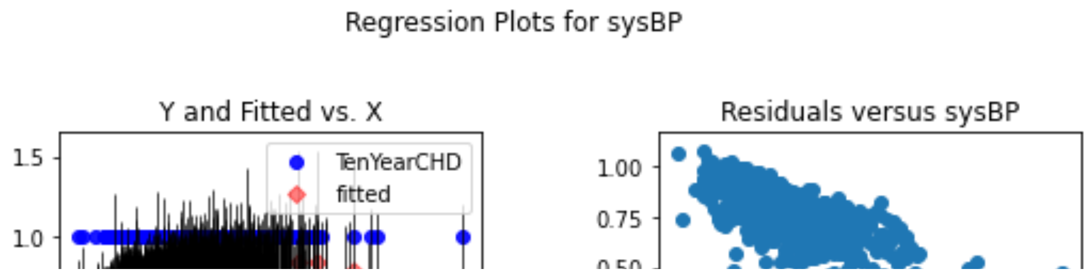
eval_env: 1

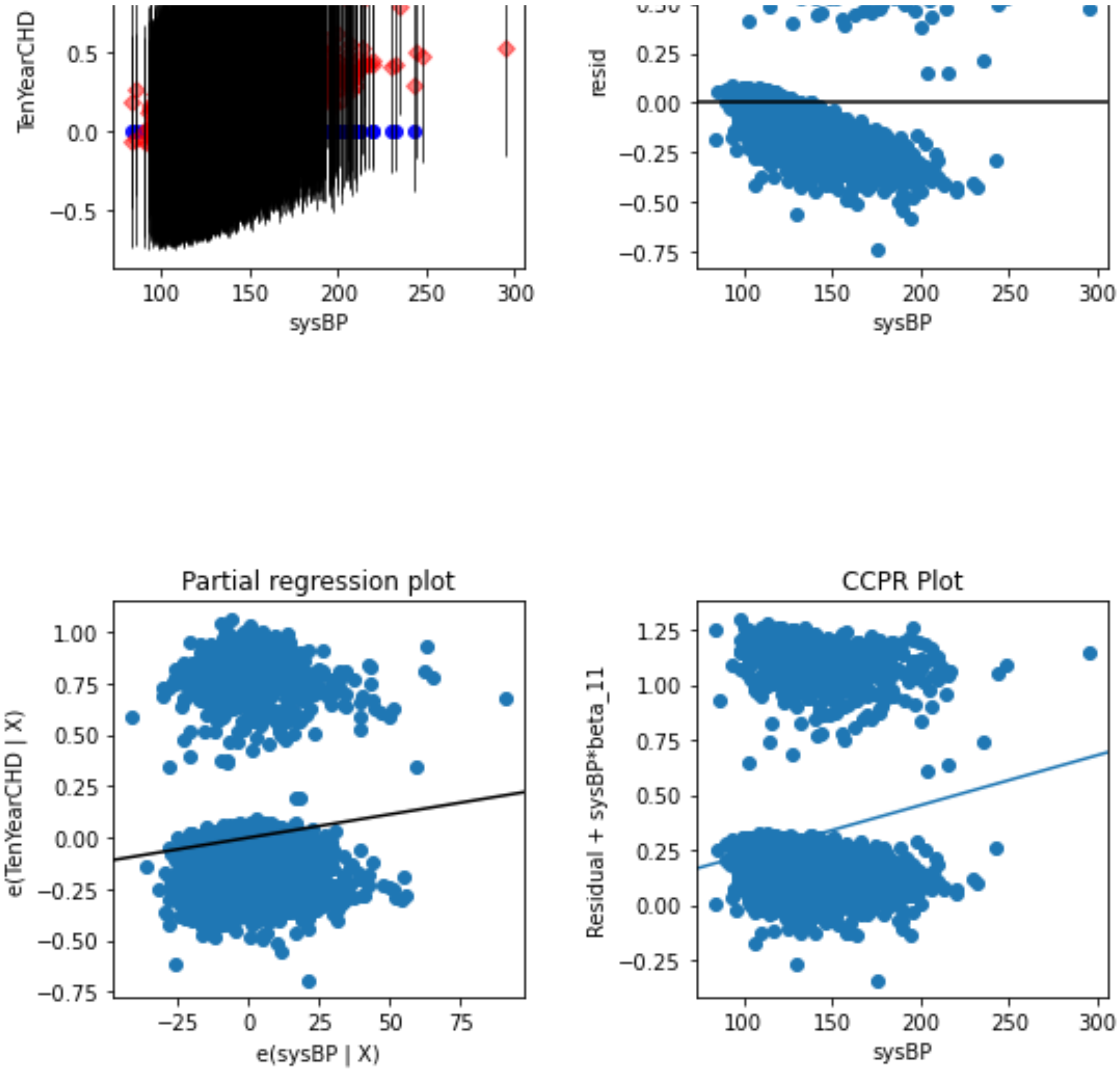




In [84]:
```
fig = sm.graphics.plot_regress_exog(result1, "sysBP")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

eval_env: 1

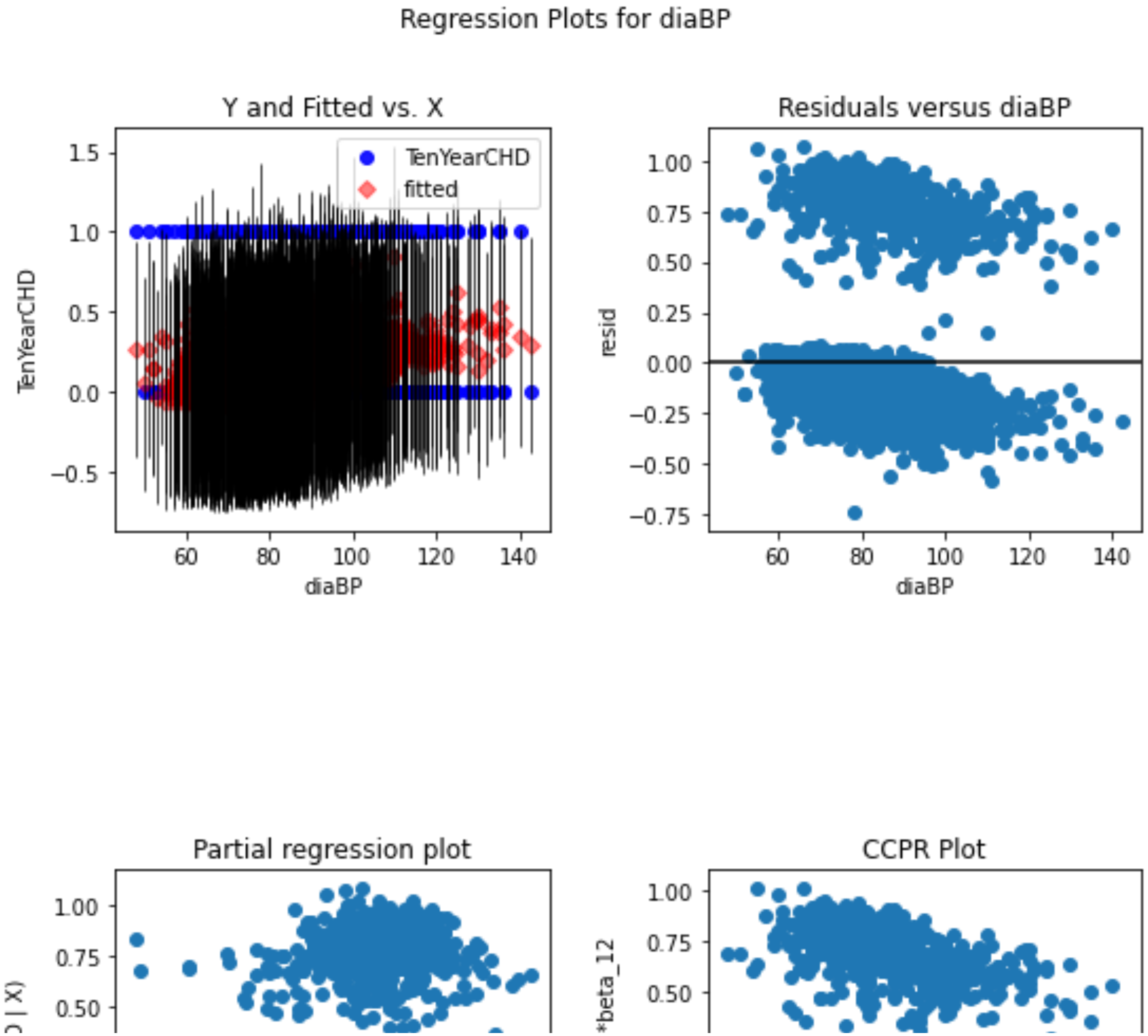


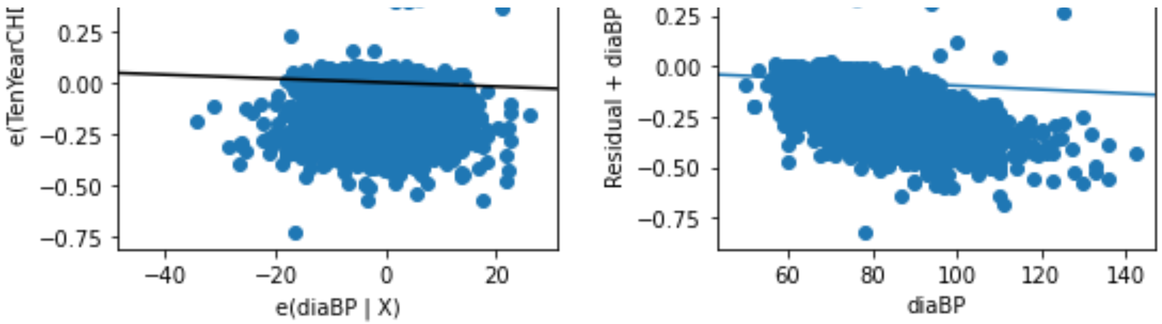

TenYearCHD
sysBP
resid
sysBP
Partial regression plot
e(TenYearCHD | X)
e(sysBP | X)
CCPR Plot
Residual + sysBP*beta_11
sysBP

In [85]:

```
fig = sm.graphics.plot_regress_exog(result1, "diaBP")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```

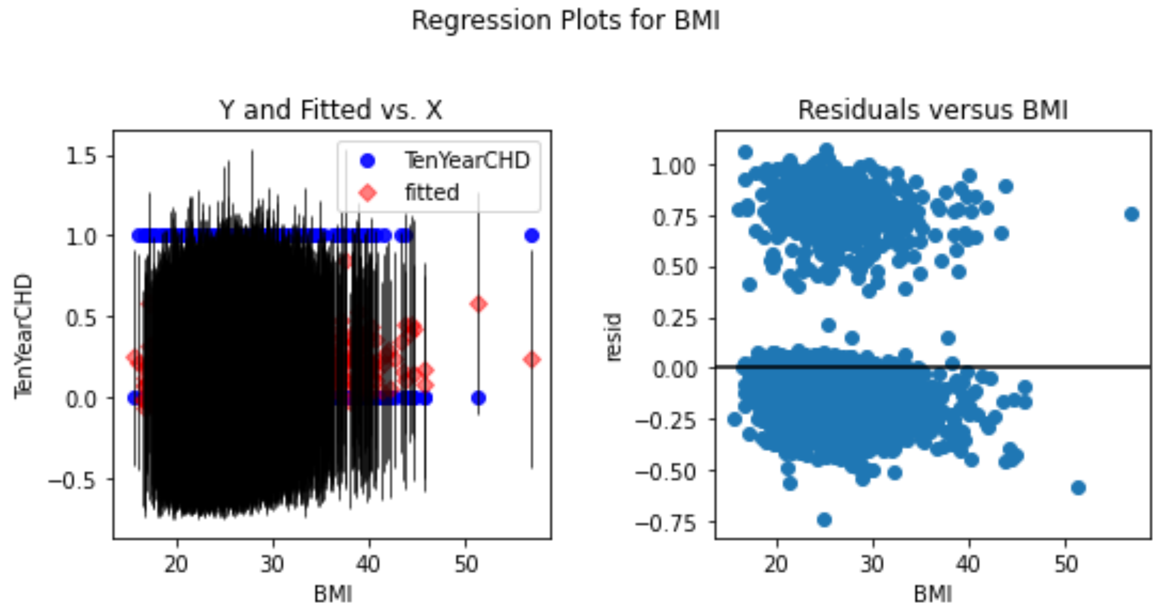


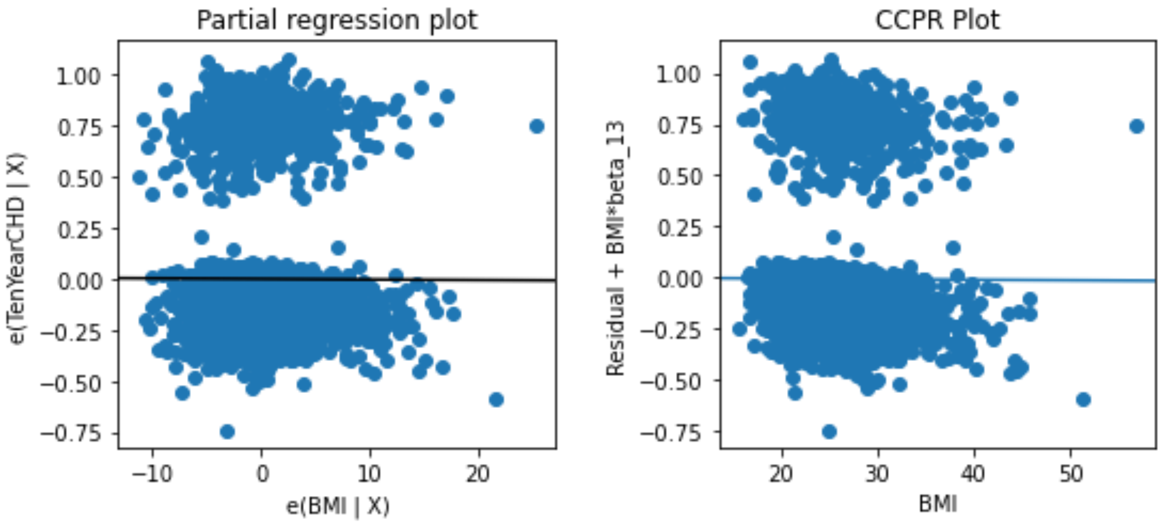


In [86]:

```
fig = sm.graphics.plot_regress_exog(result1, "BMI")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

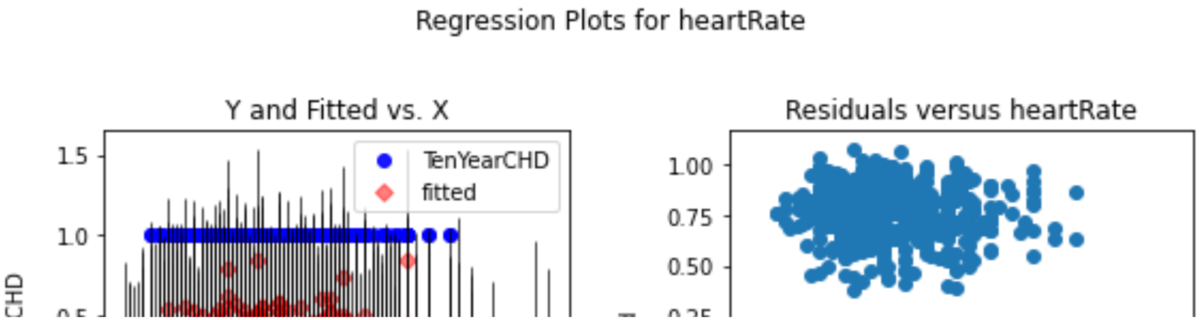
```
eval_env: 1
```

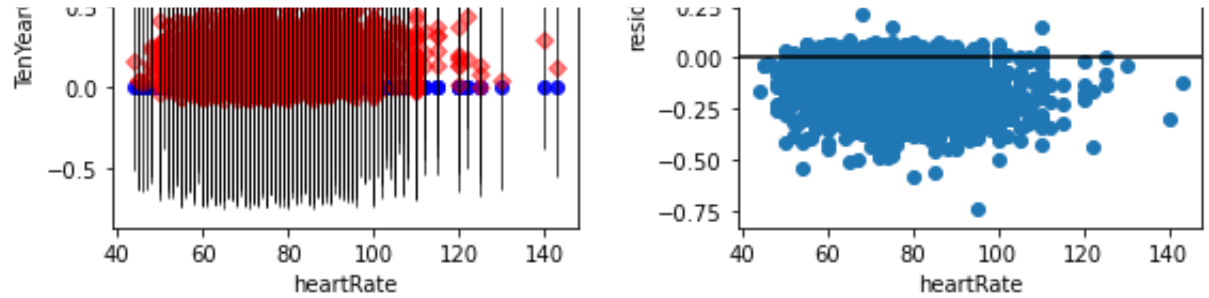


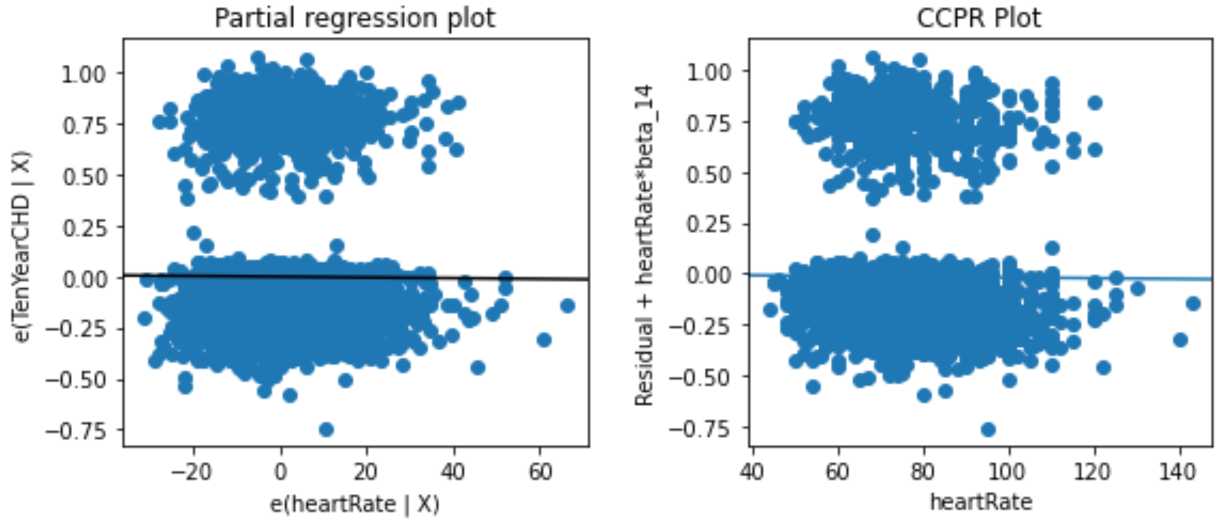


In [87]:
```
fig = sm.graphics.plot_regress_exog(result1, "heartRate")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```



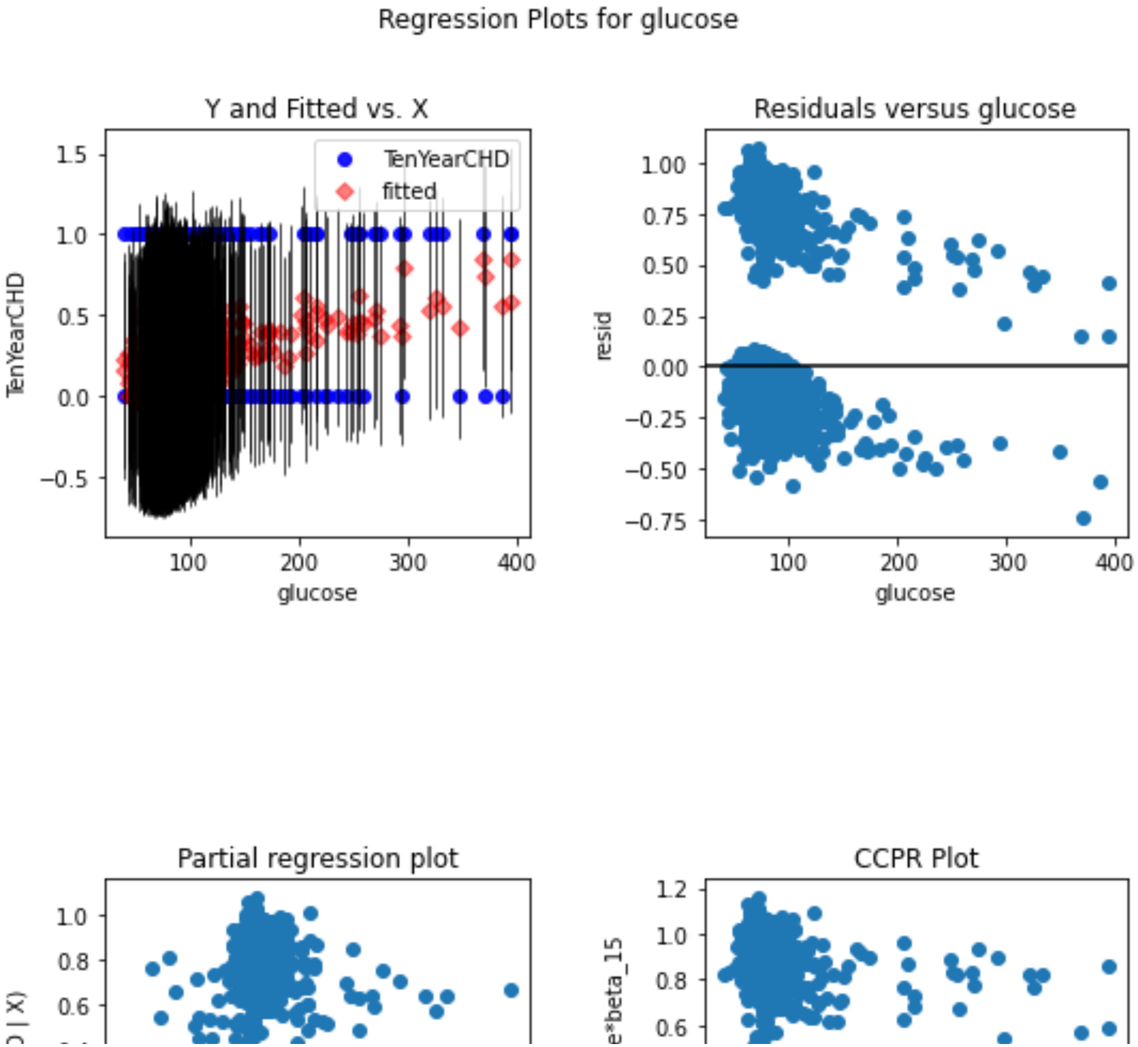

TenYear
0.5
0.0
−0.5
40
60
80
100
120
140
heartRate
resid
0.25
0.00
−0.25
−0.50
−0.75
heartRate

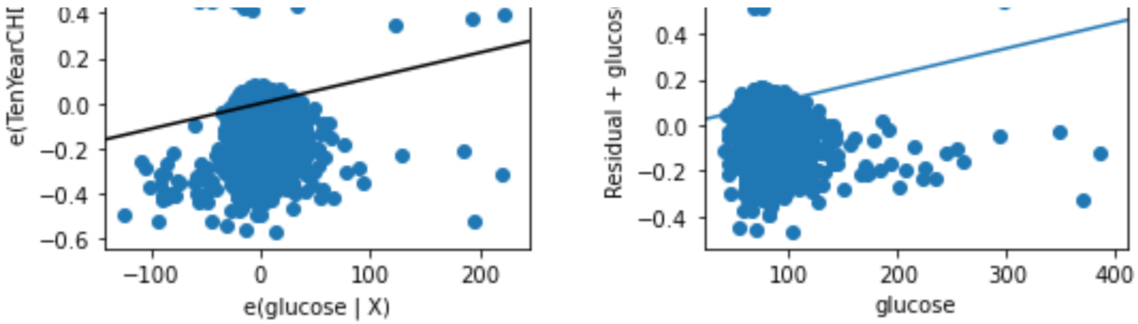

Partial regression plot
e(TenYearCHD | X)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−20
0
20
40
60
e(heartRate | X)
CCPR Plot
Residual + heartRate*beta_14
40
60
80
100
120
140
heartRate

In [88]:

```
fig = sm.graphics.plot_regress_exog(result1, "glucose")
fig.set_figheight(10)
fig.set_figwidth(8)
plt.show()
```

```
eval_env: 1
```





Logit Model

In [90]:
```
import wooldridge as woo
import statsmodels.formula.api as smf


# Estimate a logit model:
#reg_logit = smf.logit(formula='inlf ~ nwifeinc + educ + exper +I(exper**2) + age + kidslt6 + kidsge6', d
reg_logit = smf.logit(formula='TenYearCHD ~ male + age + education + currentSmoker + cigsPerDay + BPMeds

# disp = 0 avoids printing out information during the estimation:
results_logit = reg_logit.fit(disp=0)
print(f'results_logit.summary(): \n{results_logit.summary()}\n')

# log likelihood value:
print(f'results_logit.llf: {results_logit.llf}\n')

# McFadden's pseudo R2:
print(f'results_logit.prsquared: {results_logit.prsquared}\n')
```

```
results_logit.summary():
                           Logit Regression Results
==============================================================================
Dep. Variable:             TenYearCHD   No. Observations:                 4238
```

```
Model:                          Logit   Df Residuals:                     4222
Method:                           MLE   Df Model:                           15
Date:                Wed, 30 Nov 2022   Pseudo R-squ.:                  0.1115
Time:                        17:15:54   Log-Likelihood:                -1604.4
converged:                       True   LL-Null:                       -1805.8
Covariance Type:            nonrobust   LLR p-value:                 1.834e-76
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept          -8.1149      0.665    -12.201      0.000      -9.418      -6.811
male                0.5030      0.100      5.011      0.000       0.306       0.700
age                 0.0621      0.006      9.992      0.000       0.050       0.074
education          -0.0131      0.046     -0.284      0.777      -0.104       0.078
currentSmoker       0.0133      0.143      0.093      0.926      -0.267       0.293
cigsPerDay          0.0214      0.006      3.793      0.000       0.010       0.032
BPMeds              0.2435      0.220      1.105      0.269      -0.188       0.675
prevalentStroke     0.9611      0.442      2.176      0.030       0.096       1.827
prevalentHyp        0.2307      0.128      1.796      0.073      -0.021       0.483
diabetes            0.1880      0.294      0.639      0.523      -0.389       0.765
totChol             0.0018      0.001      1.780      0.075      -0.000       0.004
sysBP               0.0141      0.004      3.983      0.000       0.007       0.021
diaBP              -0.0028      0.006     -0.474      0.636      -0.015       0.009
BMI                 0.0031      0.012      0.263      0.793      -0.020       0.026
heartRate          -0.0015      0.004     -0.384      0.701      -0.009       0.006
glucose             0.0067      0.002      3.134      0.002       0.003       0.011
===================================================================================

results_logit.llf: -1604.4031293469973

results_logit.prsquared: 0.1115153761286799
```

Linear Probability Model

In [111]:
```
import wooldridge as woo
import pandas as pd
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt


# y = 1 (woman in the labor force), = 0 (otherwise)
# Estimate a linear probability model:
reg_lin = smf.ols(formula='TenYearCHD ~ male + age + education + currentSmoker + cigsPerDay + BPMeds + p
                  data=df2)
results_lin = reg_lin.fit(cov_type='HC3')

# Print regression table:
table = pd.DataFrame({'b': round(results_lin.params, 4),
                      'se': round(results_lin.bse, 4),
                      't': round(results_lin.tvalues, 4),
                      'pval': round(results_lin.pvalues, 4)})
print(f'table: \n{table}\n')

# We can check the y_hat values for two "extreme" cases:
# Recall that for this model y_hat may be outside [0, 1],
#  which in theory can't happen

X_new = pd.DataFrame(
    {'male': [1, 0], 'education': [1, 4],
     'currentSmoker': [0, 1], 'cigsPerDay': [0, 70],
     'BPMeds': [0, 1], 'prevalentStroke': [0, 1], 'prevalentHyp': [0,1], 'diabetes': [0,1], 'totChol': [
    'sysBP': [83.5,295], 'diaBP': [48,142.5], 'BMI': [15.54,56.8], 'heartRate': [44,143], 'glucose': [40
predictions = results_lin.predict(X_new)

print(f'predictions: \n{predictions}\n')
```

```
table:
                     b      se       t    pval
Intercept      -0.5592  0.0785 -7.1242  0.0000
```

```
male              0.0529  0.0118  4.4742  0.0000
age               0.0069  0.0007  9.3123  0.0000
education        -0.0018  0.0054 -0.3247  0.7454
currentSmoker    -0.0005  0.0161 -0.0289  0.9769
cigsPerDay        0.0027  0.0007  3.5956  0.0003
BPMeds            0.0550  0.0436  1.2606  0.2075
prevalentStroke   0.1947  0.0973  2.0016  0.0453
prevalentHyp      0.0288  0.0181  1.5890  0.1121
diabetes          0.0472  0.0518  0.9109  0.3623
totChol           0.0001  0.0001  0.6430  0.5202
sysBP             0.0023  0.0005  4.2675  0.0000
diaBP            -0.0010  0.0009 -1.1079  0.2679
BMI              -0.0003  0.0016 -0.1723  0.8632
heartRate        -0.0002  0.0005 -0.3828  0.7018
glucose           0.0011  0.0004  3.1355  0.0017

predictions:
0   -0.101301
1    1.425534
dtype: float64
```

Probit Model

In [92]:

```

```

```
import wooldridge as woo
import statsmodels.formula.api as smf


# Estimate a probit model:
reg_probit = smf.probit(formula='TenYearCHD ~ male + age + education + currentSmoker + cigsPerDay + BPMe
                        data=df2)
results_probit = reg_probit.fit(disp=0)
print(f'results_probit.summary(): \n{results_probit.summary()}\n')

# log likelihood value:
print(f'results_probit.llf: {results_probit.llf}\n')

# McFadden's pseudo R2:
print(f'results_probit.prsquared: {results_probit.prsquared}\n')
```

```
results_probit.summary():
                          Probit Regression Results
==============================================================================
Dep. Variable:             TenYearCHD   No. Observations:                 4238
Model:                         Probit   Df Residuals:                     4222
Method:                           MLE   Df Model:                           15
Date:                Wed, 30 Nov 2022   Pseudo R-squ.:                  0.1115
Time:                        17:20:17   Log-Likelihood:                -1604.5
converged:                       True   LL-Null:                       -1805.8
Covariance Type:            nonrobust   LLR p-value:                 1.919e-76
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept          -4.4907      0.359    -12.524      0.000      -5.193      -3.788
male                0.2617      0.055      4.786      0.000       0.155       0.369
age                 0.0342      0.003     10.020      0.000       0.028       0.041
education          -0.0114      0.025     -0.450      0.652      -0.061       0.038
currentSmoker       0.0201      0.079      0.255      0.798      -0.134       0.174
cigsPerDay          0.0118      0.003      3.683      0.000       0.006       0.018
BPMeds              0.1620      0.129      1.251      0.211      -0.092       0.416
```

```
prevalentStroke     0.5671      0.269      2.109      0.035       0.040       1.094
prevalentHyp        0.1266      0.072      1.761      0.078      -0.014       0.267
diabetes            0.1450      0.172      0.844      0.398      -0.192       0.482
totChol             0.0009      0.001      1.647      0.100      -0.000       0.002
sysBP               0.0078      0.002      3.874      0.000       0.004       0.012
diaBP              -0.0016      0.003     -0.464      0.643      -0.008       0.005
BMI                 0.0014      0.006      0.222      0.824      -0.011       0.014
heartRate          -0.0008      0.002     -0.395      0.692      -0.005       0.003
glucose             0.0036      0.001      2.951      0.003       0.001       0.006
==================================================================================

results_probit.llf: -1604.4501143428351

results_probit.prsquared: 0.1114893568402544
```

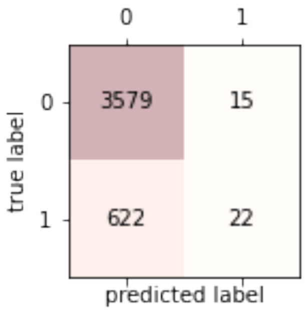
Estimating the accuracy of Logit model

In [94]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report, precision_score
import matplotlib.pyplot as plt
X_cols = ['male', 'age' , 'education' , 'currentSmoker' , 'cigsPerDay' , 'BPMeds' , 'prevalentStroke' ,
lr = LogisticRegression()
logit_mod = lr.fit(df2[X_cols], df2['TenYearCHD'])
conf_mat = confusion_matrix(df2['TenYearCHD'], lr.predict(df2[X_cols]))
print(conf_mat)
print('Accuracy =', lr.score(df2[X_cols],df2['TenYearCHD']))

# Confusion matrix plot Raschka (2014)
fig, ax = plt.subplots(figsize=(2, 2))
ax.matshow(conf_mat, cmap=plt.cm.Reds, alpha=0.3)
for i in range(conf_mat.shape[0]):
```

```
    for j in range(conf_mat.shape[1]):
        ax.text(x=j, y=i,
        s=conf_mat[i, j],
        va='center', ha='center')
plt.xlabel('predicted label')
plt.ylabel('true label')
plt.show()

# We can print other metrics
print(classification_report(df2['TenYearCHD'], lr.predict(df2[X_cols]), digits=3))

# TP = confusion[1,1] # true positive
# TN = confusion[0,0] # true negatives
# FP = confusion[0,1] # false positives
# FN = confusion[1,0] # false negatives

cm1 = confusion_matrix(df2['TenYearCHD'], lr.predict(df2[X_cols]))
total1=sum(sum(cm1))
Accuracy = (cm1[0,0]+cm1[1,1])/total1
Specificity = cm1[0,0]/(cm1[0,0]+cm1[0,1])
Sensitivity = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Accuracy =', Accuracy)
print('Specificity = ', Specificity)
print('Sensitivity = ', Sensitivity)
```

```
[[3579   15]
 [ 622   22]]
Accuracy = 0.8496932515337423

/Users/snehilshandilya/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules
/preprocessing.html)
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```




```
              precision    recall  f1-score   support

           0      0.852     0.996     0.918      3594
           1      0.595     0.034     0.065       644

    accuracy                          0.850      4238
   macro avg      0.723     0.515     0.491      4238
weighted avg      0.813     0.850     0.789      4238

Accuracy = 0.8496932515337423
Specificity =  0.9958263772954925
Sensitivity =  0.034161490683229816
```

In [103]:
```
def prs_result(model_name, model, df2):
    pred_values = model.predict(df2)
    pred_values = np.where(pred_values > 0.5, 1, 0)
    actual_values = df2['TenYearCHD']

    ACC = accuracy_score(actual_values, pred_values)
    ER = 1 - ACC
    SENS = recall_score(actual_values, pred_values)
    SPEC = recall_score(actual_values, pred_values, pos_label = 0)
    PPV = precision_score(actual_values, pred_values)
    NPV = precision_score(actual_values, pred_values, pos_label = 0)

    return pd.DataFrame([{'model': model_name, 'ACC': ACC, 'ER': ER, 'SENS': SENS, 'SPEC': SPEC, 'PPV':
```

In [102]:
```
from sklearn.metrics import recall_score
```

Estimating the Probit model

In [104]:
```
def conf_matrix(model, df2):
    pred_values = model.predict(df2)
    pred_values = np.where(pred_values > 0.5, 1, 0)
    actual_values = df2['TenYearCHD']
    return pd.crosstab(actual_values, pred_values, rownames=['Actual'], colnames=['Predicted'])
```

In [105]:
```
print("Confusion matrix for probit model\n\n{0}".format(conf_matrix(results_probit, df2)))
```

```
Confusion matrix for probit model

Predicted     0   1
Actual
0          3577  17
1           595  49
```

In [106]:
```
precision_models_score_probit = prs_result('Probit Model', results_probit, df2)
precision_models_score_probit
```

Out[106]:

	model	ACC	ER	SENS	SPEC	PPV	NPV
0	Probit Model	0.85559	0.14441	0.07609	0.99527	0.74242	0.85738

Estimating the Linear Probability Model

In [112]:
```
print("Confusion matrix for linear probability model\n\n{0}".format(conf_matrix(results_lin, df2)))
```

```
Confusion matrix for linear probability model

Predicted     0   1
Actual
0          3589   5
1           621  23
```

In [114]:
```
precision_models_score_linear = prs_result('Linear Model', results_lin, df2)
precision_models_score_linear
```

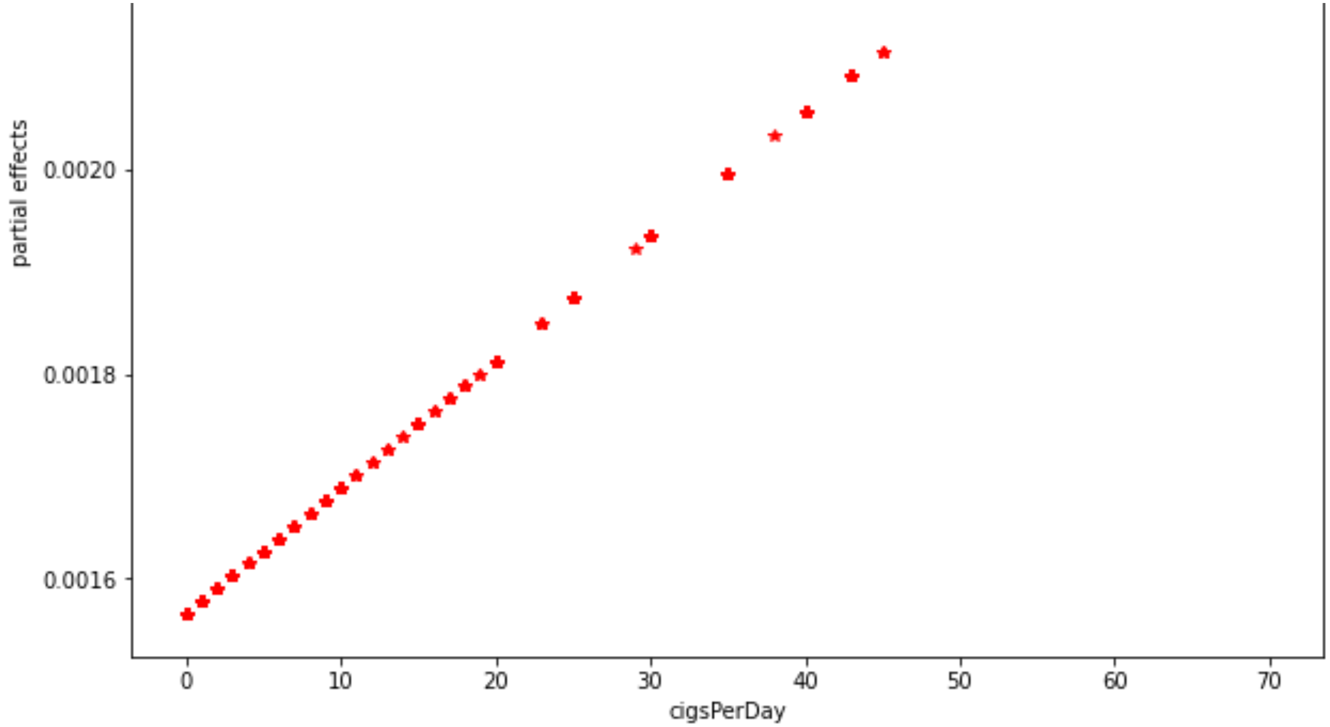
Out[114]:

	model	ACC	ER	SENS	SPEC	PPV	NPV
0	Linear Model	0.85229	0.14771	0.03571	0.99861	0.82143	0.85249

In [117]:

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
import scipy.stats as stats

#estimation
reg_probit2 = smf.probit(formula = 'TenYearCHD ~ cigsPerDay', data = df2)
results_probit2 = reg_probit2.fit(disp = 0)

# calculate partial effects:
xb_probit = results_probit2.fittedvalues
factor_probit = stats.norm.pdf(xb_probit)
PE_probit = results_probit2.params['cigsPerDay'] * factor_probit


# plot APE's:
x = df2['cigsPerDay']
fig, ax = plt.subplots(figsize=(10, 8))

plt.plot(x, PE_probit, color='red',
         marker='*', linestyle='', label='probit')
plt.ylabel('partial effects')
plt.xlabel('cigsPerDay')
plt.legend()
```

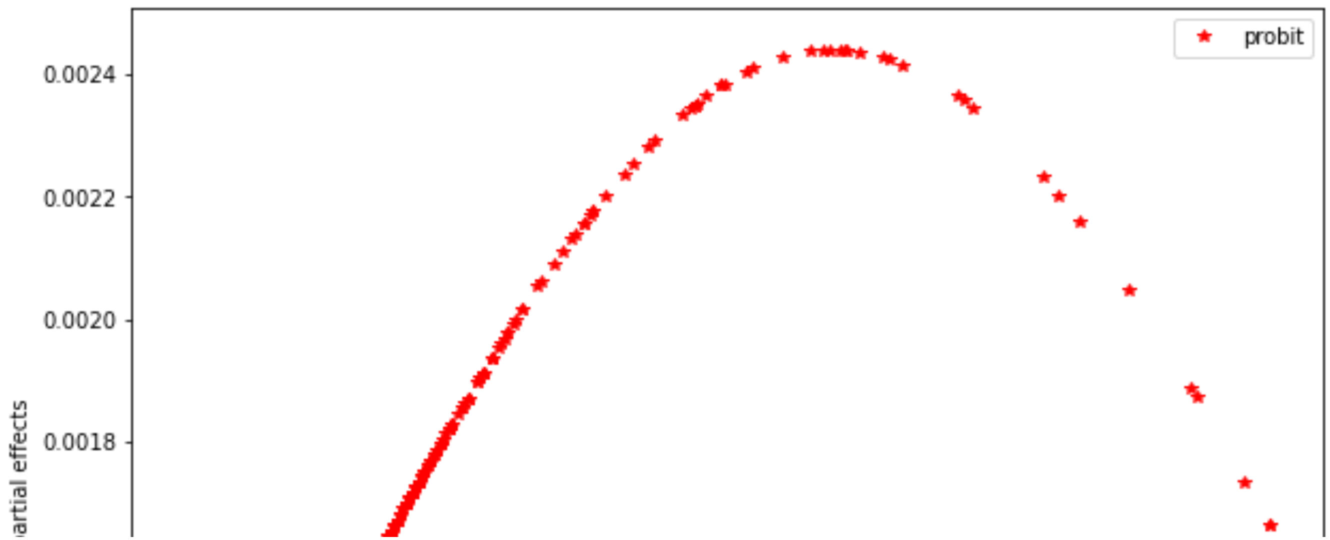
Out[117]: <matplotlib.legend.Legend at 0x7ff21684c5b0>

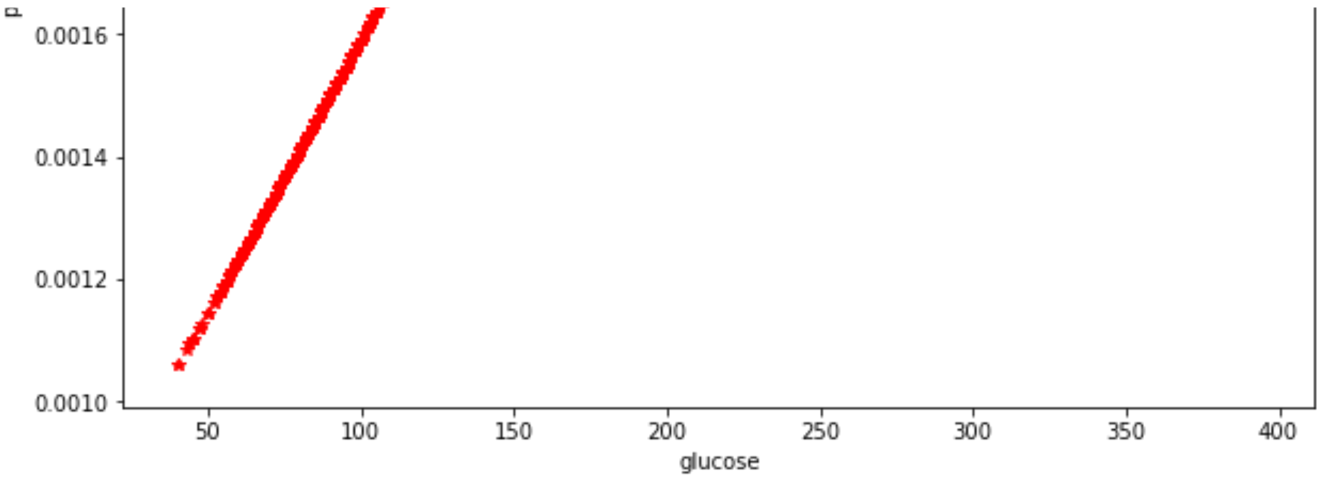

partial effects
0.0020
0.0018
0.0016
0
10
20
30
40
50
60
70
cigsPerDay

In [118]:

```
reg_probit2 = smf.probit(formula = 'TenYearCHD ~ glucose', data = df2)
results_probit2 = reg_probit2.fit(disp = 0)

# calculate partial effects:
xb_probit = results_probit2.fittedvalues
factor_probit = stats.norm.pdf(xb_probit)
PE_probit = results_probit2.params['glucose'] * factor_probit


# plot APE's:
x = df2['glucose']
fig, ax = plt.subplots(figsize=(10, 8))

plt.plot(x, PE_probit, color='red',
         marker='*', linestyle='', label='probit')
plt.ylabel('partial effects')
plt.xlabel('glucose')
plt.legend()
```

Out[118]: <matplotlib.legend.Legend at 0x7ff1e15380a0>

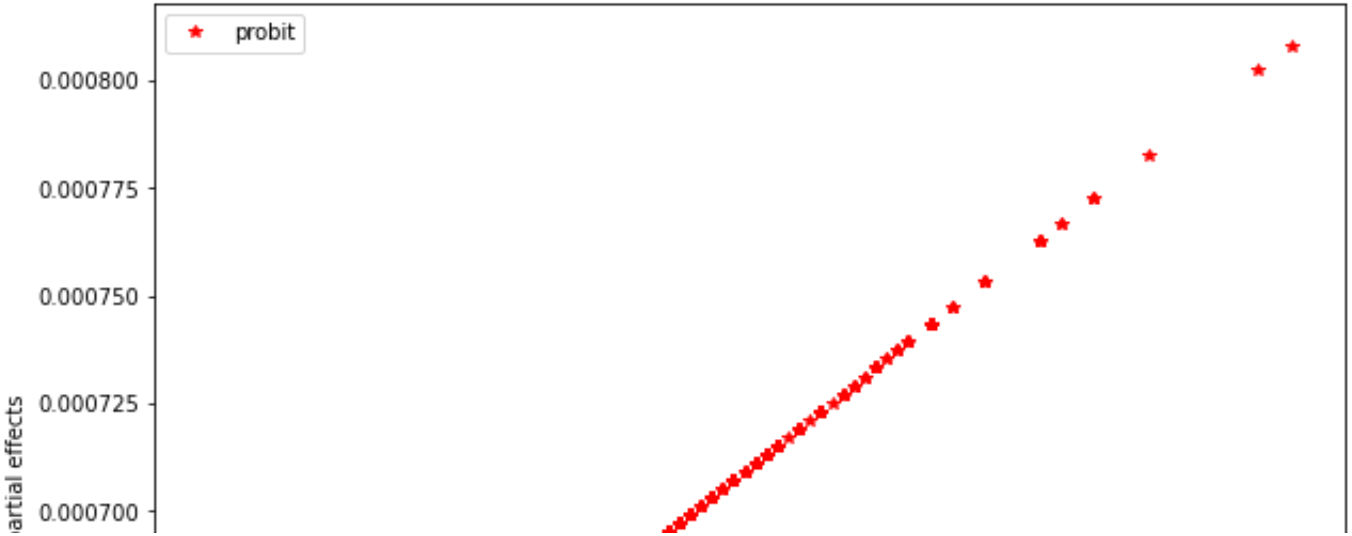


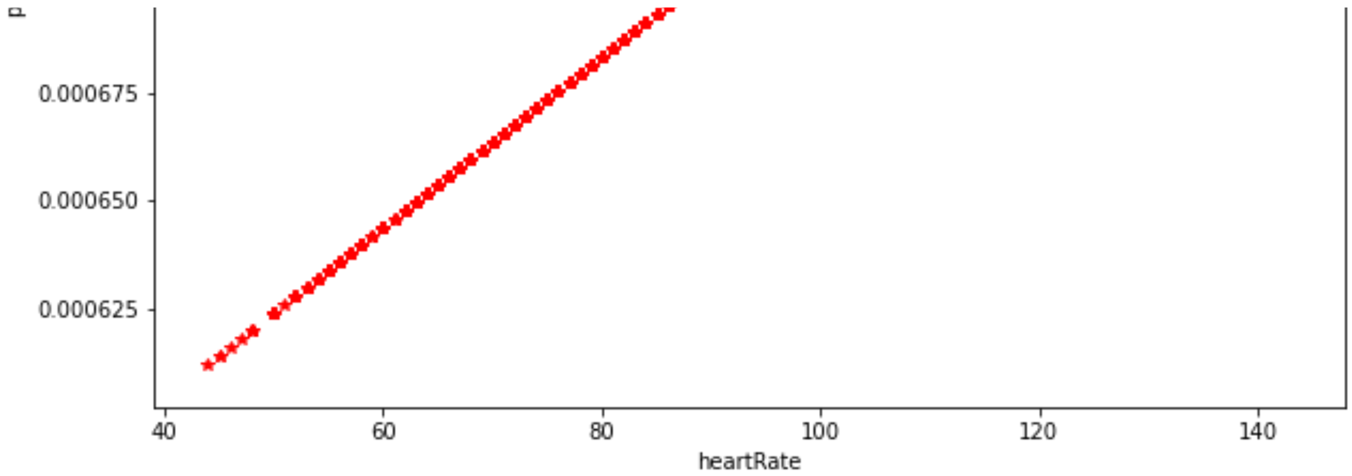

p
0.0016
0.0014
0.0012
0.0010
50
100
150
200
250
300
350
400
glucose

In [119]:

```
reg_probit2 = smf.probit(formula = 'TenYearCHD ~ heartRate', data = df2)
results_probit2 = reg_probit2.fit(disp = 0)

# calculate partial effects:
xb_probit = results_probit2.fittedvalues
factor_probit = stats.norm.pdf(xb_probit)
PE_probit = results_probit2.params['heartRate'] * factor_probit


# plot APE's:
x = df2['heartRate']
fig, ax = plt.subplots(figsize=(10, 8))

plt.plot(x, PE_probit, color='red',
         marker='*', linestyle='', label='probit')
plt.ylabel('partial effects')
plt.xlabel('heartRate')
plt.legend()
```

Out[119]: <matplotlib.legend.Legend at 0x7ff216f95f10>

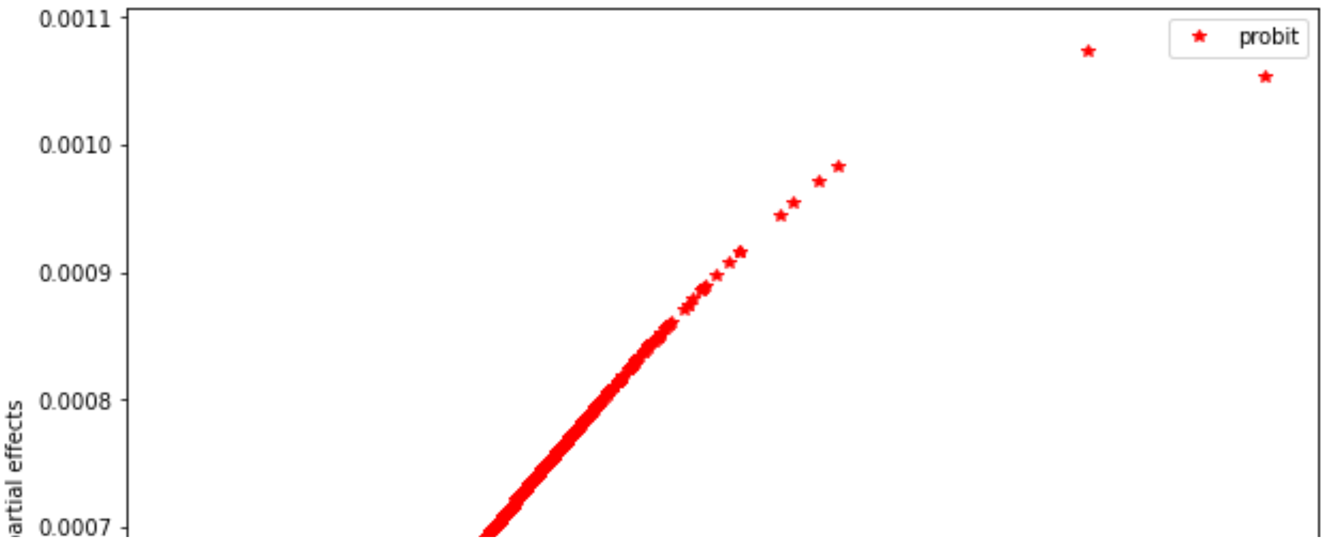


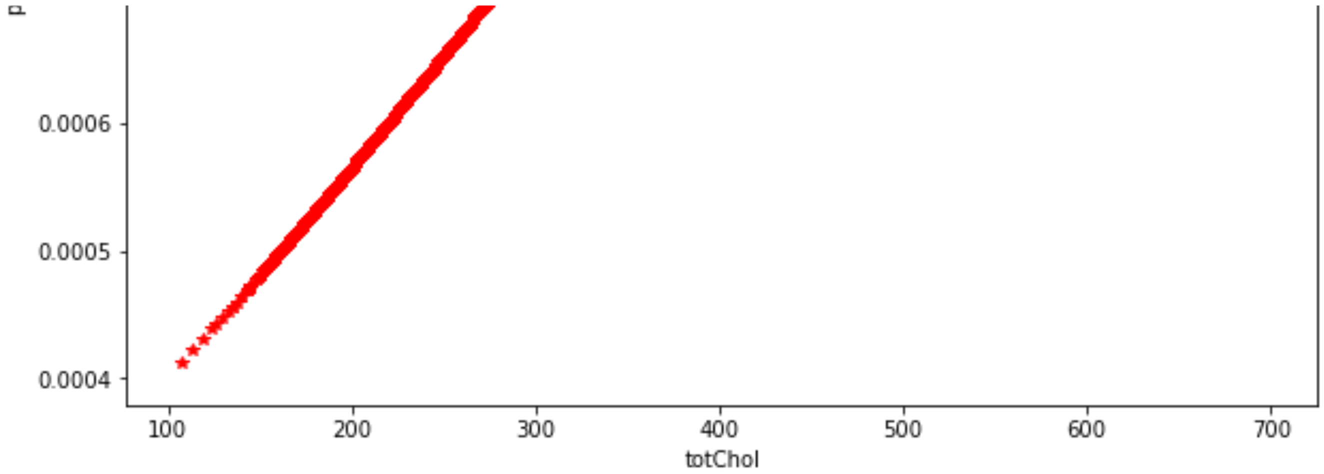


In [120]:

```
reg_probit2 = smf.probit(formula = 'TenYearCHD ~ totChol', data = df2)
results_probit2 = reg_probit2.fit(disp = 0)

# calculate partial effects:
xb_probit = results_probit2.fittedvalues
factor_probit = stats.norm.pdf(xb_probit)
PE_probit = results_probit2.params['totChol'] * factor_probit


# plot APE's:
x = df2['totChol']
fig, ax = plt.subplots(figsize=(10, 8))

plt.plot(x, PE_probit, color='red',
         marker='*', linestyle='', label='probit')
plt.ylabel('partial effects')
plt.xlabel('totChol')
plt.legend()
```

Out[120]: <matplotlib.legend.Legend at 0x7ff1d8981ac0>





In [ ]: