In [7]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
```

In [8]:
```
#with 1 feature
df=pd.read_csv("f:/dataset/regression/salary_poly.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10)

model=LinearRegression()
model.fit(X_train,y_train)

print(model.score(X_train,y_train))
print(model.score(X_test,y_test))

#Result-->model is underfit
```

```
0.7853524430203845
0.5233724904068862
```

In [9]:
```
#Solution---->need more features
             #Polynomial fetaures are one of the techniques

pf=PolynomialFeatures(degree=2)
X_train_new=pf.fit_transform(X_train)
X_test_new=pf.transform(X_test)

model=LinearRegression()
model.fit(X_train_new,y_train)

print(model.score(X_train_new,y_train))
print(model.score(X_test_new,y_test))

#Result--->getting low bias than 1 feature but high variance
#solution->we need to tune degree
```

```
0.8437695501667827
0.4222293235966541
```

In [21]:
```
pf=PolynomialFeatures(degree=9)
X_train_new=pf.fit_transform(X_train)
X_test_new=pf.transform(X_test)

model=LinearRegression()
model.fit(X_train_new,y_train)

print(model.score(X_train_new,y_train))
print(model.score(X_test_new,y_test))
```

```
0.9500903759555972
0.9137104148422517
```

In [22]:
```
from sklearn.datasets import load_diabetes
```

In [23]:
```
dib=load_diabetes()
```

In [24]:
```
print(dib.DESCR)
```

```
.. _diabetes_dataset:

Diabetes dataset
```

```
---------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

  :Number of Instances: 442

  :Number of Attributes: First 10 columns are numeric predictive values

  :Target: Column 11 is a quantitative measure of disease progression one year after bas
eline

  :Attribute Information:
      - age     age in years
      - sex
      - bmi     body mass index
      - bp      average blood pressure
      - s1      tc, total serum cholesterol
      - s2      ldl, low-density lipoproteins
      - s3      hdl, high-density lipoproteins
      - s4      tch, total cholesterol / HDL
      - s5      ltg, possibly log of serum triglycerides level
      - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the stand
ard deviation times the square root of `n_samples` (i.e. the sum of squares of each colu
mn totals 1).

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle R
egression," Annals of Statistics (with discussion), 407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```

In [25]:
```
dib.target
```

Out[25]:
```
array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 310., 101.,
        69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
        68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
        87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
       259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
       128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
       150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
       200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
        42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
        83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
       104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
       173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
       107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
```

```
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
       220.,  57.])
```

In [26]:
```
X=dib.data
y=dib.target
```

In [27]:
```
X.shape
```

Out[27]:
```
(442, 10)
```

In [29]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10)

model=LinearRegression()
model.fit(X_train,y_train)

print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.5112345828164674
0.5282320385429605
```

In [42]:
```
pf=PolynomialFeatures(degree=1)
X_train_new=pf.fit_transform(X_train)
X_test_new=pf.transform(X_test)

model=LinearRegression()
model.fit(X_train_new,y_train)

print(model.score(X_train_new,y_train))
print(model.score(X_test_new,y_test))
```

```
0.5112345828164674
0.5282320385429603
```

In [ ]:
```
#Assumptions of LinearRegression
1. No multicolinearity.
2. Homoscedasticity of residuals.
3. Each feature should be normally distributed.
4. Each feature represents linear relationship with target.
5. samples>features
```

In [45]:
```
print(model.predict(X_test_new[:10]))
```

```
[148.09033248 204.20923318 184.3462547   82.45321738 163.69963569
 126.16907295 125.99461797 262.17609803  80.64638978  74.27657335]
```

In [47]:
```
print(y_train[:10])
```

```
[174. 214. 202.  99.  94. 235. 158.  59. 220. 275.]
```

In [48]:
```
174-148
```

Out[48]:
```
26
```

In [49]:
```
204-214
```

Out[49]:
```
-10
```

In [50]:
```
model.predict(X_test_new[:10])-y_train[:10]
```

Out[50]:
```
array([ -25.90966752,   -9.79076682,  -17.6537453 ,  -16.54678262,
         69.69963569, -108.83092705,  -32.00538203,  203.17609803,
       -139.35361022, -200.72342665])
```

In [51]:
```
df=pd.DataFrame(X_train,columns=dib.feature_names)
df
```

Out[51]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	-0.005515	-0.044642	0.008883	-0.050427	0.025950	0.047224	-0.043401	0.071210	0.014821	0.003064
1	-0.063635	-0.044642	-0.033151	-0.033213	0.001183	0.024051	-0.024993	-0.002592	-0.022517	-0.059067
2	0.005383	0.050680	0.030440	0.083844	-0.037344	-0.047347	0.015505	-0.039493	0.008641	0.015491
3	-0.089063	-0.044642	-0.061174	-0.026328	-0.055231	-0.054549	0.041277	-0.076395	-0.093937	-0.054925
4	-0.092695	0.050680	-0.090275	-0.057313	-0.024960	-0.030437	-0.006584	-0.002592	0.024055	0.003064
...	...	...	...	...	...	...	...	...	...	...
326	-0.009147	-0.044642	0.037984	-0.040099	-0.024960	-0.003819	-0.043401	0.015858	-0.005142	0.027917
327	-0.023677	-0.044642	0.030440	-0.005670	0.082364	0.092004	-0.017629	0.071210	0.033043	0.003064
328	-0.052738	0.050680	-0.018062	0.080401	0.089244	0.107662	-0.039719	0.108111	0.036060	-0.042499
329	-0.005515	0.050680	-0.008362	-0.002228	-0.033216	-0.063630	-0.036038	-0.002592	0.080590	0.007207
330	-0.034575	0.050680	-0.025607	-0.017135	0.001183	-0.002880	0.008142	-0.015508	0.014821	0.040343

331 rows × 10 columns

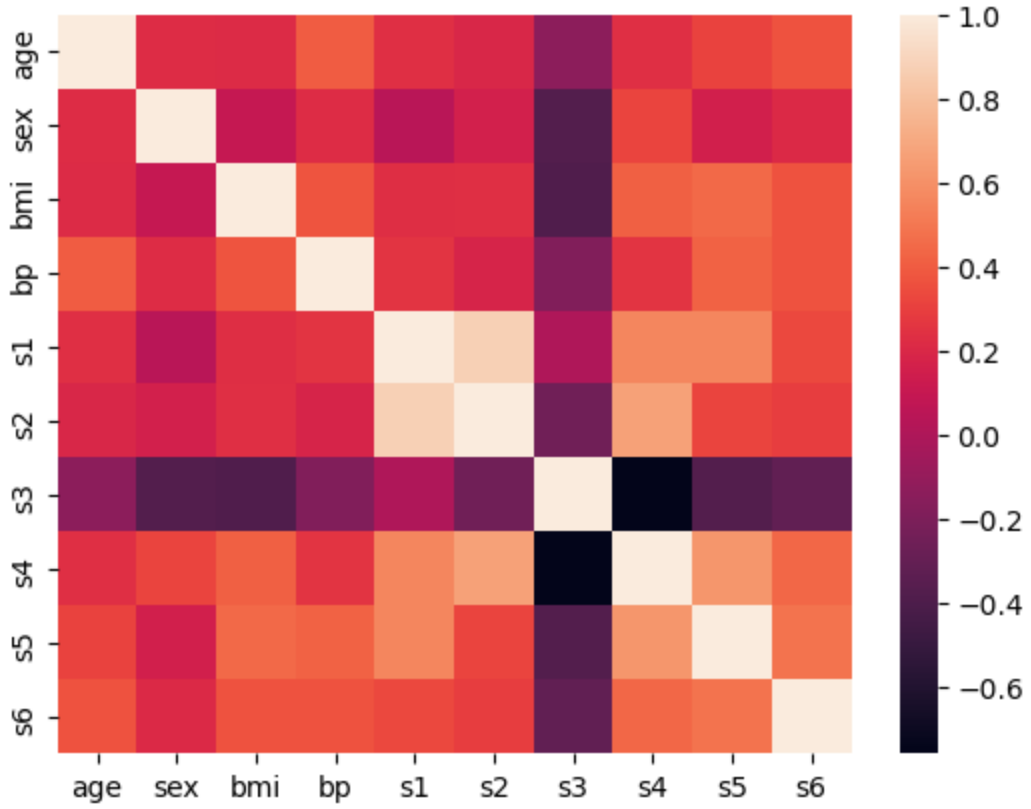
In [52]:
```
df.corr()
```

Out[52]:

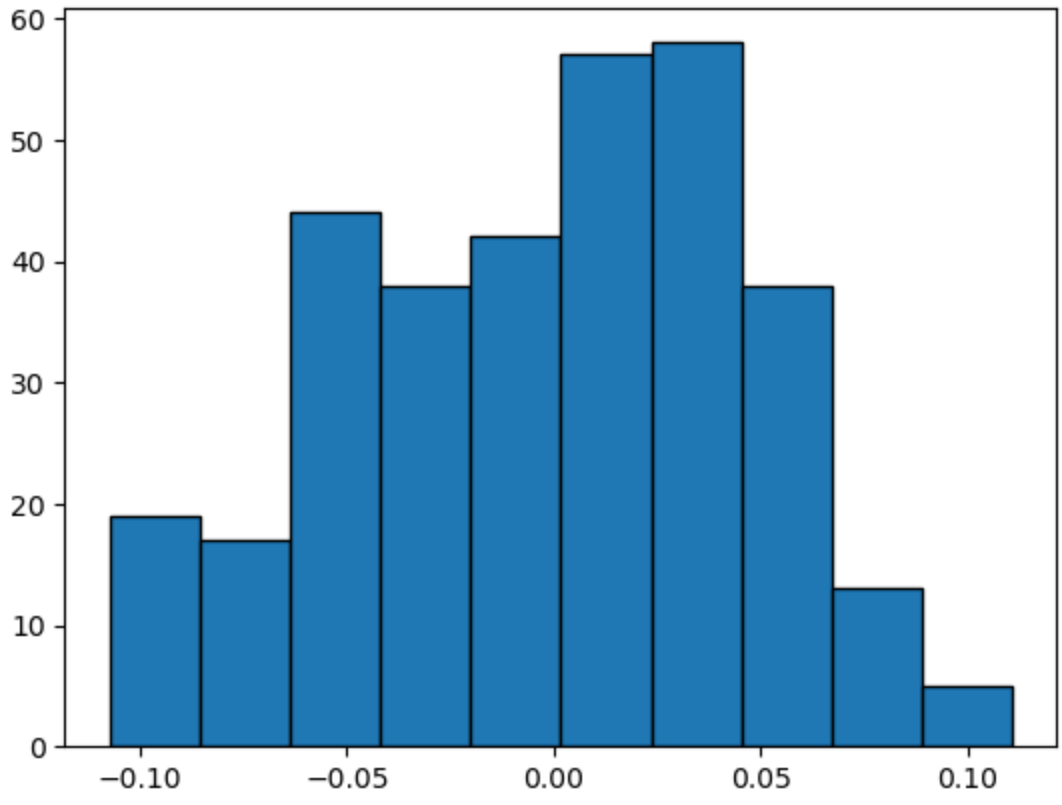
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
age	1.000000	0.219252	0.215665	0.397598	0.235799	0.193022	-0.136845	0.234589	0.307258	0.363364
sex	0.219252	1.000000	0.095095	0.223241	0.044155	0.158547	-0.376109	0.316405	0.148389	0.207810
bmi	0.215665	0.095095	1.000000	0.371090	0.227401	0.236289	-0.390043	0.412333	0.445155	0.366618
bp	0.397598	0.223241	0.371090	1.000000	0.254582	0.177349	-0.187339	0.252787	0.417100	0.367498
s1	0.235799	0.044155	0.227401	0.254582	1.000000	0.874376	0.002859	0.553710	0.559923	0.331571
s2	0.193022	0.158547	0.236289	0.177349	0.874376	1.000000	-0.255726	0.663623	0.316288	0.289919
s3	-0.136845	-0.376109	-0.390043	-0.187339	0.002859	-0.255726	1.000000	-0.758258	-0.379101	-0.311971
s4	0.234589	0.316405	0.412333	0.252787	0.553710	0.663623	-0.758258	1.000000	0.616762	0.443108
s5	0.307258	0.148389	0.445155	0.417100	0.559923	0.316288	-0.379101	0.616762	1.000000	0.483125
s6	0.363364	0.207810	0.366618	0.367498	0.331571	0.289919	-0.311971	0.443108	0.483125	1.000000

In [54]:
```
import matplotlib.pyplot as plt
import seaborn as sb
```

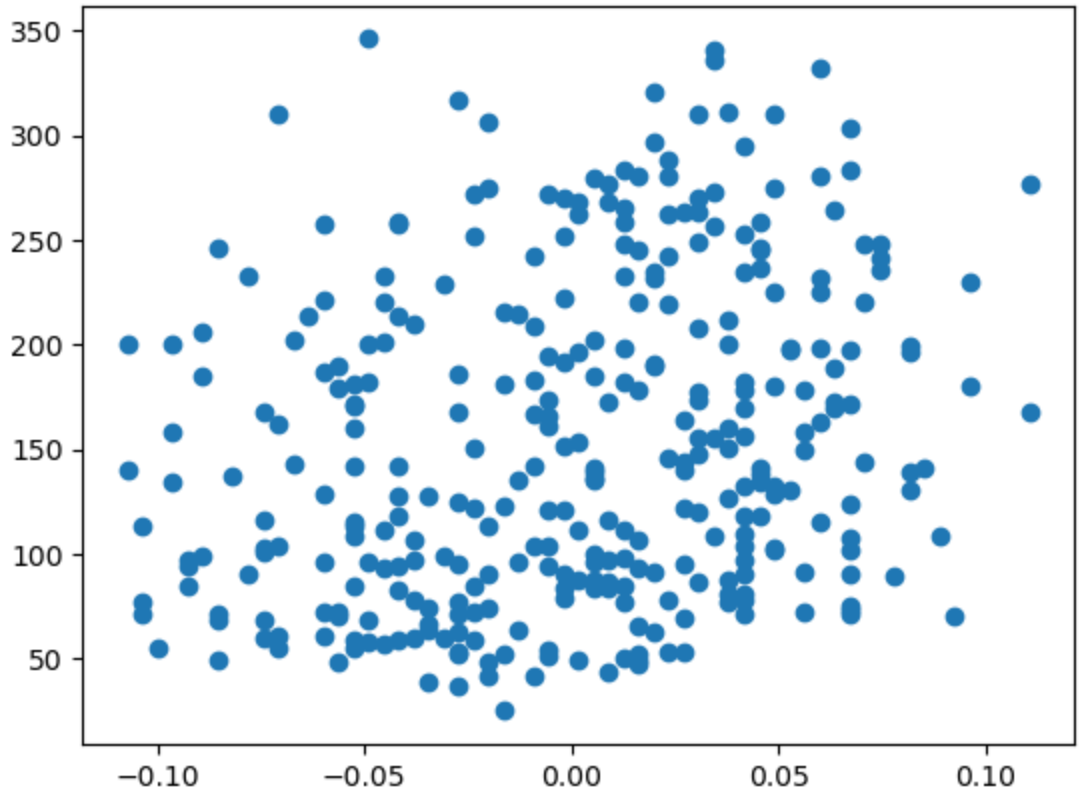
In [55]:
```
sb.heatmap(df.corr())
plt.show()
```



In [67]:
```
plt.hist(df.age,edgecolor='k')
plt.show()
```



In [80]:
```
plt.scatter(df.age,y_train)
plt.show()
```



In [ ]: