In [7]:

```
# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# Where to save the figures
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "pictures"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR,CHAPTER_ID)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=600):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)

# Ignore useless warnings
import warnings
warnings.filterwarnings(action="ignore", message="^internal gelsd")
```

Get the data

datasets_link: https://www.kaggle.com/kumarajarshi/life-expectancy-who/code (https://www.kaggle.com/kumarajarshi/life-expectancy-who/code)

dataset file has been downloaded in folder called archive

In [8]:

```
import pandas as pd

path = os.getcwd()
DATASET_PATH = os.path.join(path,"archive")
def load_data(in_path):
    csv_path = os.path.join(DATASET_PATH,in_path)
    return pd.read_csv(csv_path)
```

In [9]:

```
df = load_data("Life Expectancy Data.csv")
df.head()
```

Out[9]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0

5 rows × 22 columns

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
Country                            2938 non-null object
Year                               2938 non-null int64
Status                             2938 non-null object
Life expectancy                    2928 non-null float64
Adult Mortality                    2928 non-null float64
infant deaths                      2938 non-null int64
Alcohol                            2744 non-null float64
percentage expenditure             2938 non-null float64
Hepatitis B                        2385 non-null float64
Measles                            2938 non-null int64
 BMI                               2904 non-null float64
under-five deaths                  2938 non-null int64
Polio                              2919 non-null float64
Total expenditure                  2712 non-null float64
Diphtheria                         2919 non-null float64
 HIV/AIDS                          2938 non-null float64
GDP                                2490 non-null float64
Population                         2286 non-null float64
 thinness  1-19 years              2904 non-null float64
 thinness 5-9 years                2904 non-null float64
Income composition of resources    2771 non-null float64
Schooling                          2775 non-null float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

We can find that right columns are all have 2938 rows,but it's obsulute not true!

So we're supported to fix this data and remove some useless datas to make our predict module better!

The processing measure is to delete all the data of that row if it encounters an empty cell

In [14]:

```
import csv

path_now = os.path.join(DATASET_PATH,"Life Expectancy Data.csv")
f = open(path_now,mode="r",encoding="utf-8")
csv_reader = csv.reader(f)
rows = [i for i in csv_reader]
# Get the file's headers
headers = rows[0]
# Make new csv file to save right datas
fp = open(DATASET_PATH + '/' + "data_end.csv",mode="w+",newline="",encoding="utf-8-sig")
csv_write = csv.writer(fp)
csv_write.writerow(headers)
for row in rows[1:]:
    flag = True
    for j in range(len(row)):
        if row[j].strip() == "":
            flag = False
            break
    if flag:
        csv_write.writerow(row)
fp.close()
```

In [15]:

```
data = load_data("data_end.csv")
data.head()
```

Out[15]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263	62	0.01	71.279624	65
1	Afghanistan	2014	Developing	59.9	271	64	0.01	73.523582	62
2	Afghanistan	2013	Developing	59.9	268	66	0.01	73.219243	64
3	Afghanistan	2012	Developing	59.5	272	69	0.01	78.184215	67
4	Afghanistan	2011	Developing	59.2	275	71	0.01	7.097109	68

5 rows × 22 columns

In [16]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1649 entries, 0 to 1648
Data columns (total 22 columns):
Country                            1649 non-null object
Year                               1649 non-null int64
Status                             1649 non-null object
Life expectancy                    1649 non-null float64
Adult Mortality                    1649 non-null int64
infant deaths                      1649 non-null int64
Alcohol                            1649 non-null float64
percentage expenditure             1649 non-null float64
Hepatitis B                        1649 non-null int64
Measles                            1649 non-null int64
 BMI                               1649 non-null float64
under-five deaths                  1649 non-null int64
Polio                              1649 non-null int64
Total expenditure                  1649 non-null float64
Diphtheria                         1649 non-null int64
 HIV/AIDS                          1649 non-null float64
GDP                                1649 non-null float64
Population                         1649 non-null float64
 thinness  1-19 years              1649 non-null float64
 thinness 5-9 years                1649 non-null float64
Income composition of resources    1649 non-null float64
Schooling                          1649 non-null float64
dtypes: float64(12), int64(8), object(2)
memory usage: 283.5+ KB
```

In [17]:

```
data.describe()
```

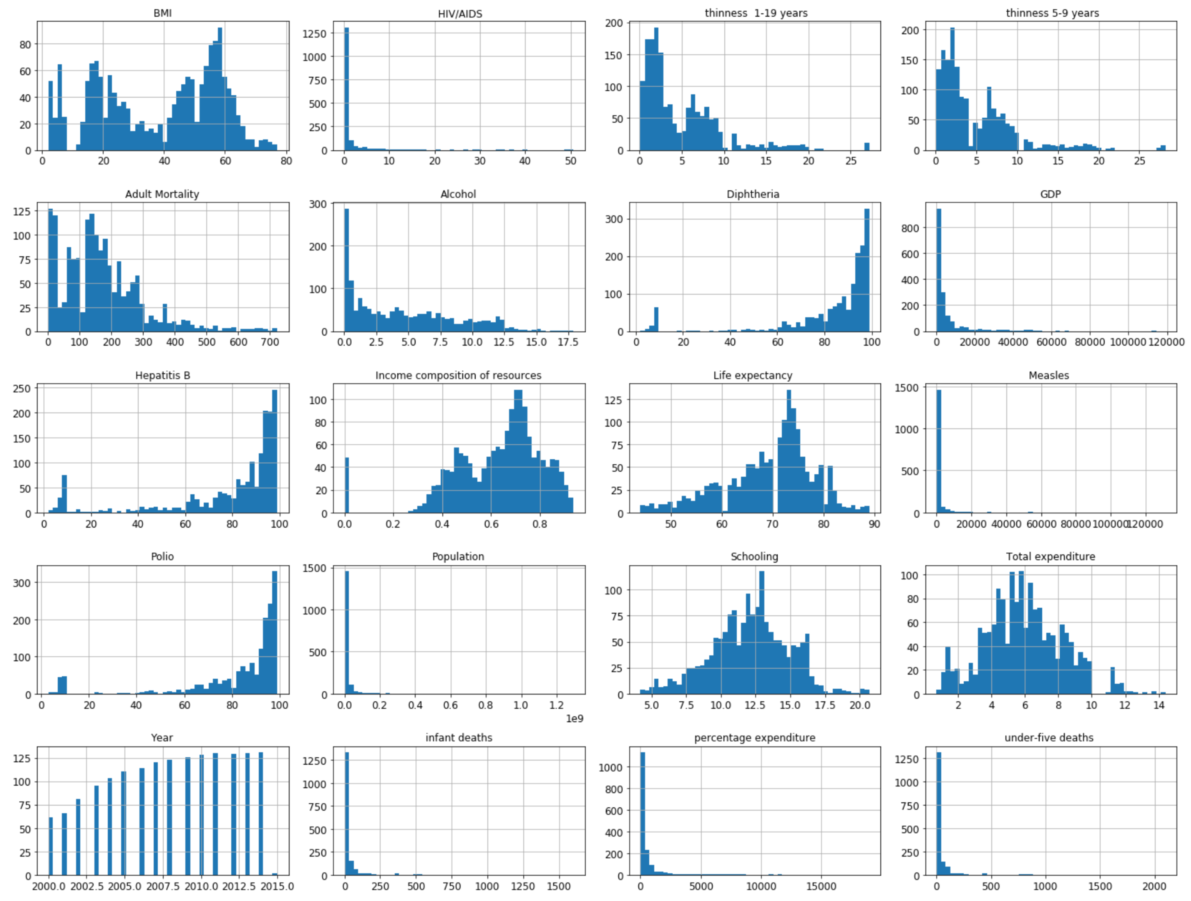
Out[17]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepati
count	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.00
mean	2007.840509	69.302304	168.215282	32.553062	4.533196	698.973558	79.21
std	4.087711	8.796834	125.310417	120.847190	4.029189	1759.229336	25.60
min	2000.000000	44.000000	1.000000	0.000000	0.010000	0.000000	2.00
25%	2005.000000	64.400000	77.000000	1.000000	0.810000	37.438577	74.00
50%	2008.000000	71.700000	148.000000	3.000000	3.790000	145.102253	89.00
75%	2011.000000	75.000000	227.000000	22.000000	7.340000	509.389994	96.00
max	2015.000000	89.000000	723.000000	1600.000000	17.870000	18961.348600	99.00

In [18]:

```
data.hist(bins=50, figsize=(20,15))
save_fig("attribute_histogram_plots")
plt.show()
```

```
Saving figure attribute_histogram_plots
```



1、The various predictors initially selected include demographic variables, income composition and mortality

```
According to actual life experience, life expectancy is the result, and other variables except
resource income composition are also the result. Therefore, only the resource income composition
of the initial choice will really affect life expectancy!
```

Use multiple linear regression to explore the relationship between resource income composition and life expectancy

In [28]:

```
# Here is a reference to cross-validation
from sklearn.model_selection import train_test_split
# Linear regression
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

In [39]:

```
X = data.loc[:,('Adult Mortality','infant deaths','under-five deaths ','Income composition of resour
y = data.loc[:,'Life expectancy ']
X_train,X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2,random_state=100)
print('X_train.shape={}\ny_train.shape ={}\nX_test.shape={}\ny_test.shape={}'.format(X_train.shape,y
linreg = LinearRegression()
model = linreg.fit(X_train, y_train)
print(model)
# Model intercept after training
print(linreg.intercept_)
# Model weight after training (no change in the number of features)
print(linreg.coef_)
```

```
X_train.shape=(1319, 4)
y_train.shape =(1319,)
X_test.shape=(330, 4)
y_test.shape=(330,)
LinearRegression()
59.845035425471885
[-0.03159116  0.0903467  -0.07140856 23.71559925]
```

In [40]:

```
feature_cols = ['Adult Mortality','infant deaths','under-five deaths ','Income composition of resour
B = list(zip(feature_cols,linreg.coef_))
print(B)
```

```
[('Adult Mortality', -0.03159116192145392), ('infant deaths', 0.090346702981388894),
('under-five deaths ', -0.07140856329713798), ('Income composition of resources', 2
3.715599250377643)]
```

In [42]:

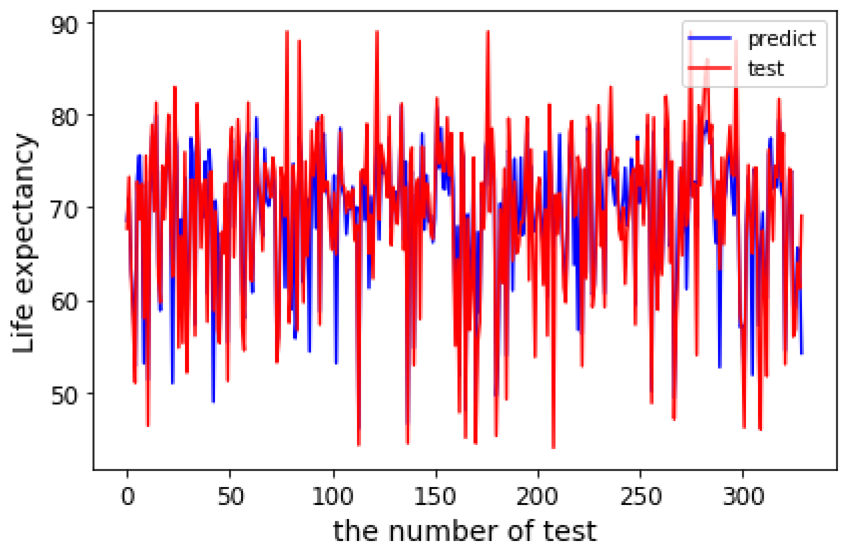
```python
# predict
y_pred = linreg.predict(X_test)
print(y_pred)
```

```
[68.55631781 71.00955094 63.30391865 59.85836686 52.81675124 73.13137241
 75.62458469 70.19887045 53.09442483 72.8088162  51.29416414 69.67394025
 77.6810332  71.71631202 79.91489533 63.87415637 58.82497257 73.88657672
 70.00905561 74.72853498 78.03703889 70.9754978  50.9552659  78.50355925
 76.64541856 58.76675818 68.68910667 65.88329537 73.187126   55.17557189
 64.42874768 77.4741347  75.84572282 57.2815791  79.34121406 73.48769608
 68.8217852  68.95831222 74.98206178 60.640179   76.2399677  72.90898952
 48.92760817 70.72795459 67.01394143 56.73887272 67.13929028 67.02523296
 70.9791724  55.32673149 74.81270766 77.30954808 67.83493314 75.07390389
 75.04480224 72.59546472 60.76590268 57.97208282 76.486044   78.03102166
 64.5630768  60.78928009 69.4461736  79.66207712 71.86057676 69.02099436
 66.86709933 76.36083137 70.6483909  70.14133977 71.52897404 75.05798873
 71.85718916 53.80536557 58.58445272 71.7293069  71.8887742  61.3186321
 77.78988342 61.64235412 58.94782987 74.75596091 55.80765182 58.51987962
 77.59272265 75.5859193  61.85279979 67.51911883 70.92967498 54.37976441
 73.6219552  75.43601191 67.7375145  79.66491346 58.71478613 76.81833908
 77.92464714 72.0081524  71.21190658 69.59517966 65.64591881 73.49268136
 53.11611794 71.5705708  78.54894549 71.89780561 71.1336066  67.08443304
 69.74631688 69.90720676 72.25601532 68.30800763 70.0375362  46.09707557
 71.32924977 74.06335615 68.61824542 76.53844044 61.26357957 71.19625545
 66.17599704 72.73616613 77.23088137 66.50756092 74.83676169 73.75699561
 73.69443567 72.89368295 77.53927307 67.84418518 71.26562672 71.68173998
 68.73302284 73.42401412 80.9583817  66.43260972 74.97963234 46.59173832
 72.53026345 73.86743628 54.75369848 71.38420472 70.97061882 69.44697464
 77.9434914  67.58741706 73.50276862 68.43003937 69.12069482 66.21024518
 68.81348457 80.75759433 74.23912369 78.56496335 72.0804684  71.94304347
 77.49936676 71.35484587 76.78903587 71.98859644 57.74928358 67.94936218
 51.76722257 76.59975563 75.26412582 48.06869167 69.24831148 59.40656534
 66.03352562 74.76527646 48.90800014 67.33261041 58.56965508 71.8658588
 68.41384639 76.5663962  77.92040814 72.36541083 75.61875409 73.08158115
 49.57141454 54.0994734  70.43594749 63.40011112 72.05976199 53.92535658
 76.17023125 75.121424   60.99299784 75.27964655 68.32129185 67.18124765
 75.40309091 67.01650383 69.71359589 78.08756322 68.61448757 74.47289037
 67.33778471 67.89003401 70.15509414 70.84715113 68.08318818 65.91503654
 76.01960758 60.34698637 77.96783934 71.07846657 65.73038618 70.11111172
 68.35272581 77.86960215 67.97617929 64.1205338  60.88714414 70.7928285
 76.56737806 77.80306494 63.78072876 70.55621822 56.7175104  72.49237111
 56.2763728  74.0987337  62.88005202 78.60798269 76.78789381 60.55567898
 61.62675707 70.54568185 79.1198981  74.01944883 70.70423899 60.66107533
 74.08812489 73.49595117 77.30945917 72.20478029 70.27214345 73.79655125
 69.01371173 68.32065788 70.69283073 74.29474783 68.0343968  70.86762792
 75.27955763 73.96742681 59.43619006 77.62731305 70.51282489 72.69857552
 68.81222303 72.10844024 78.80213427 68.79001736 50.09784061 78.13145176
 63.4082199  61.60501942 73.96893831 63.21519704 69.93853119 78.52179107
 75.88573297 65.76450677 68.67496979 49.31905665 60.1788845  69.17600498
 74.0082985  67.17708779 77.07301447 61.16025392 70.04064677 77.79395458
 71.23870019 72.77590629 67.87628489 78.92062336 73.44755077 77.75262952
 78.15570083 79.29201956 76.99168042 77.38114441 70.4981407  66.16668397
 71.82084386 52.67181776 72.24281887 70.34324297 71.74914794 75.15294004
 76.91496975 71.63792769 69.19007335 78.40860795 71.49558899 57.01685122
 57.21085936 50.21562903 67.89983741 73.36861348 68.88161586 51.83981844
 74.16856983 72.15993035 57.1989634  65.65469655 69.46018134 62.05321548
 60.19039609 75.73848793 77.44997454 72.05467664 72.31105535 72.24521948
 79.45952532 71.83408982 70.98520571 54.45957496 65.83725853 73.40683928
 73.92353827 59.23294453 59.13611388 65.63840885 64.14717327 54.19021744]
```

In [45]:

```
sum_mean = 0
for i in range(len(y_pred)):
    sum_mean+=(y_pred[i]-y_test.values[i])**2
sum_erro = np.sqrt(sum_mean/10)
# calculate RMSE by hand
print("RMSE by hand:",sum_erro)
# Make ROC curve
plt.figure()
plt.plot(range(len(y_pred)),y_pred,'b',label="predict")
plt.plot(range(len(y_pred)),y_test,'r',label="test")
# Show labels in the figure
plt.legend(loc="upper right")
plt.xlabel("the number of test")
plt.ylabel('Life expectancy')
save_fig("linreg_plots")
plt.show()
```

```
RMSE by hand: 25.929372753292174
Saving figure linreg_plots
```



2、Should countries with life expectancy values below 65 increase their healthcare facilities to improve their average life expectancy?

```
The variables involved in health care expenditures include hepatitis B, measles, polio, HIV/AIDS
Formulate regression model based on mixed effects model and multiple linear regression
```

In [49]:

```
X = data.loc[:,('Hepatitis B','Measles ','Polio',' HIV/AIDS')]
y = data.loc[:,'Life expectancy ']
X_train,X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2,random_state=100)
print('X_train.shape={}\ny_train.shape ={}\nX_test.shape={}\ny_test.shape={}'.format(X_train.shape,y
linreg = LinearRegression()
model = linreg.fit(X_train, y_train)
print(model)
# Model intercept after training
print(linreg.intercept_)
# Model weight after training (no change in the number of features)
print(linreg.coef_)
```

```
X_train.shape=(1319, 4)
y_train.shape =(1319,)
X_test.shape=(330, 4)
y_test.shape=(330,)
LinearRegression()
62.04643543450354
[ 8.14201482e-03 -5.90352347e-05  1.00216685e-01 -7.89114194e-01]
```

In [50]:

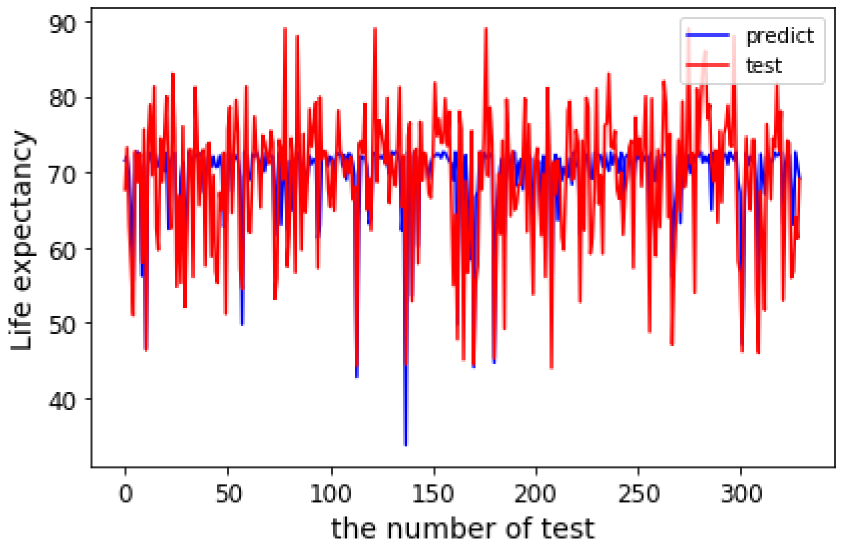
```
feature_cols = ['Hepatitis B','Measles ','Polio',' HIV/AIDS']
B = list(zip(feature_cols,linreg.coef_))
print(B)
# predict
y_pred = linreg.predict(X_test)
print(y_pred)
```

```
[('Hepatitis B', 0.008142014816139383), ('Measles ', -5.903523468947557e-05), ('Po
lio', 0.10021668541228466), (' HIV/AIDS', -0.7891141935439198)]
[71.53160368 72.10383418 69.89528157 64.94946249 55.60029911 72.36169915
 72.69503534 72.1883224  56.24509539 70.94880924 46.56428951 72.68689332
 72.44574988 71.21689124 72.46078447 70.65814262 71.11983687 72.46362301
 71.925938   70.1425702  72.57617321 62.48241877 71.5610023  72.36181722
 72.58579111 61.06084601 61.84652171 69.41614154 71.39473094 58.61403574
 66.75558147 72.69503534 71.97289341 70.81658766 72.35986906 72.31824437
 72.47831794 72.39940652 67.38877651 67.29133713 71.86575503 72.58667664
 70.78642968 72.07561883 70.71989788 70.74902394 72.18268912 70.70338901
 62.78492948 63.42647046 72.45371479 72.50525649 67.47941457 72.4310946
 71.8281067  71.64212499 69.33348945 49.80489601 72.69503534 72.12019697
 69.0121533  69.06969886 72.39120547 72.16952587 72.68653911 71.66914443
 69.58229213 72.5866176  70.10505697 72.17583551 72.15052622 72.2287373
 72.04546637 61.61371656 68.905116   72.50432249 63.04106099 69.16236064
 72.43481382 62.41083637 71.46660101 64.902817   70.52993459 66.55852896
 72.55716387 72.57764909 70.45003146 70.94127928 69.64876268 69.19339007
 72.15324184 71.11036491 71.69979018 71.61856324 61.28128244 62.94275232
 72.58573207 72.69414981 72.34529705 70.85270129 72.06374398 71.77117163
 69.42426881 72.4770782  72.41318182 72.69503534 70.85171711 72.27917297
 69.00210716 72.47011689 72.36995924 71.22876608 69.58932077 42.79854337
 63.39055935 72.15324184 71.96597172 71.92808725 70.59618632 63.17953645
 63.88282665 72.47814083 72.58667664 72.08370181 72.15324184 71.82816574
 72.26047402 71.97913497 70.95993832 69.82593346 72.69444499 72.36153979
 72.69491727 72.59027294 71.23670067 62.29896273 65.86349738 33.78553827
 72.69503534 72.25345852 53.60322253 72.6949763  72.04488314 60.11584746
 72.05790609 71.76169123 72.13695781 70.84106259 69.08229636 71.12734124
 72.08426744 72.16521145 72.45771464 72.37739283 71.88700126 72.6896041
 72.55410858 72.04488314 71.48674657 71.20242667 68.92380042 72.69361849
 49.8387502  71.60330632 71.10799112 53.66028773 72.27381922 67.08013305
 58.43051932 71.46117058 44.1126851  66.73018564 67.48155092 72.69432691
 69.59600799 72.58573207 72.1357766  71.7827464  72.47831794 72.35343907
 44.69757406 64.80679898 67.989478   70.78780132 71.91650758 61.81452188
 72.26160054 72.46911329 68.96410361 72.47784566 72.6886005  68.22376671
 71.16498268 71.7183489  67.72775571 72.07827284 71.57890725 72.26160054
 71.80814888 61.14444675 70.35046792 71.21058159 71.41299616 62.58284607
 72.18268912 68.40051423 72.11302864 70.44603718 50.98809071 70.82031034
 72.31339349 63.63476454 71.86101837 68.93867797 70.65074584 69.77631341
 69.35159517 71.88504177 68.38767982 72.69456306 70.97783519 70.74845961
 61.75393687 66.24878835 69.99532115 72.15484943 72.50525649 69.32823194
 69.13533381 72.59458251 72.35880643 72.3204951  72.15324184 70.7637977
 72.58667664 70.99888543 70.19464322 72.67060929 71.04221315 72.25345852
 71.88515984 69.96320394 72.58667664 63.79277578 71.9680714  72.28336194
 72.56944804 71.26073721 68.1837006  71.37844691 72.45015906 71.74925432
 72.59475962 71.49751599 72.17072021 72.16640509 57.15961681 72.67060929
 70.04663009 68.77684861 72.04488314 68.92005327 72.36181722 72.3990975
 72.16138385 72.585614   72.29104782 56.12357824 64.93226896 67.41678783
 72.24158368 63.25317056 70.96041061 68.15530608 70.30184591 72.22863812
 69.9088179  72.30592536 71.38325897 72.05834226 72.25334045 72.68110787
 71.2833024  72.0945899  71.42729899 72.31394272 65.01282664 72.65432526
 71.32648045 72.66995991 72.58549593 71.40258799 72.35326196 72.37509045
 71.05619965 72.6555998  72.16267226 71.60743394 72.6379822  68.88751706
 67.37087324 47.16502132 63.26146675 72.5706481  70.93798202 72.10312831
 71.56008374 71.83624872 68.60886966 54.53699082 72.42725985 69.63290095
 67.11337739 72.1530057  71.98425486 71.3380645  71.59945542 72.6154155
 72.21851491 72.37733379 72.68630297 63.10648799 71.10609985 72.57853462
 72.2615415  67.85846197 63.05197015 72.68689332 70.87414448 69.2352857 ]
```

In [51]:

```
sum_mean = 0
for i in range(len(y_pred)):
    sum_mean+=(y_pred[i]-y_test.values[i])**2
sum_erro = np.sqrt(sum_mean/10)
# calculate RMSE by hand
print("RMSE by hand:",sum_erro)
# Make ROC curve
plt.figure()
plt.plot(range(len(y_pred)),y_pred,'b',label="predict")
plt.plot(range(len(y_pred)),y_test,'r',label="test")
# Show labels in the figure
plt.legend(loc="upper right")
plt.xlabel("the number of test")
plt.ylabel('Life expectancy')
save_fig("ill_linreg_plots")
plt.show()
```

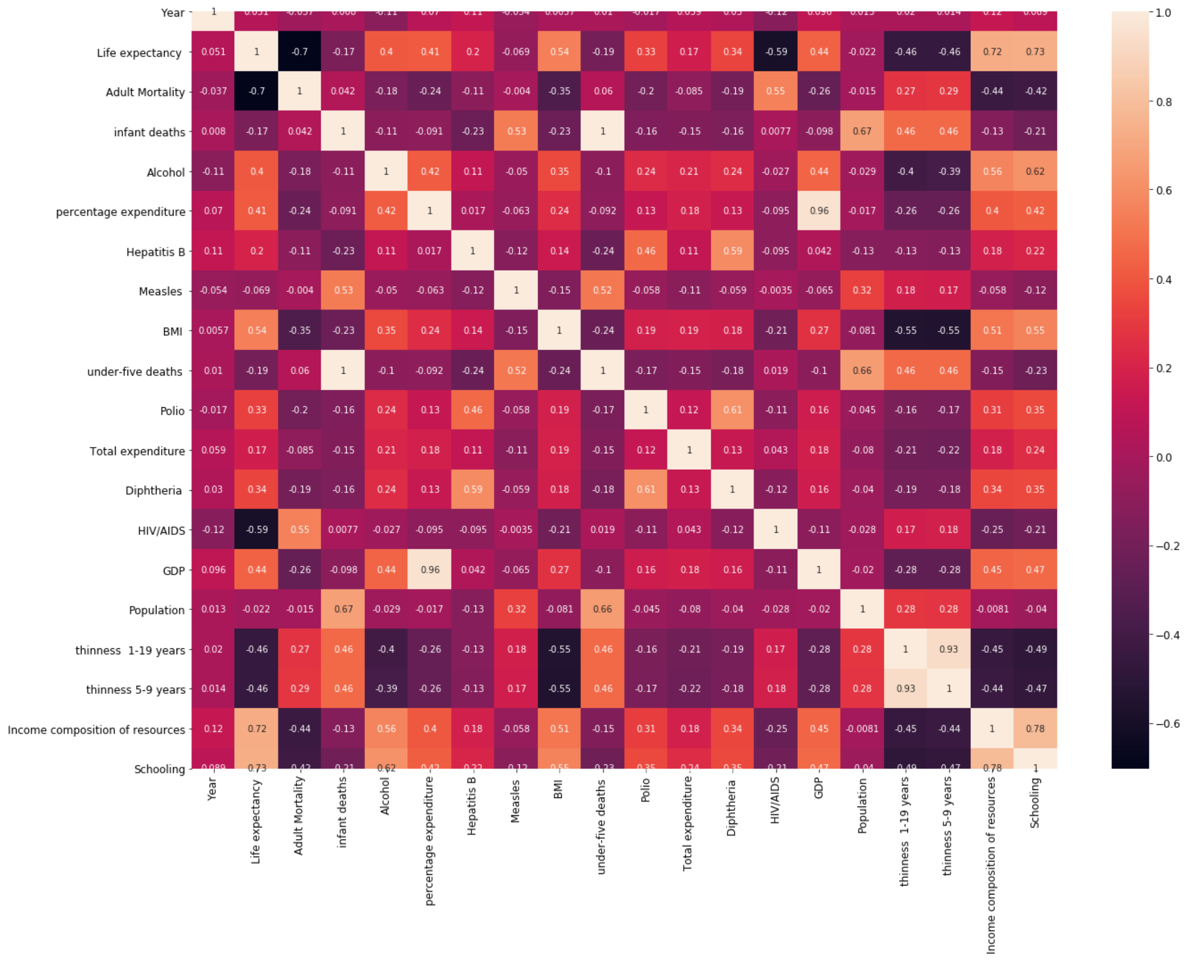
```
RMSE by hand: 37.710642084958565
Saving figure ill_linreg_plots
```



In [60]:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize = (20,15))
sns.heatmap(data.corr(),annot = True)
save_fig("axes_pic")
```

```
Saving figure axes_pic
```

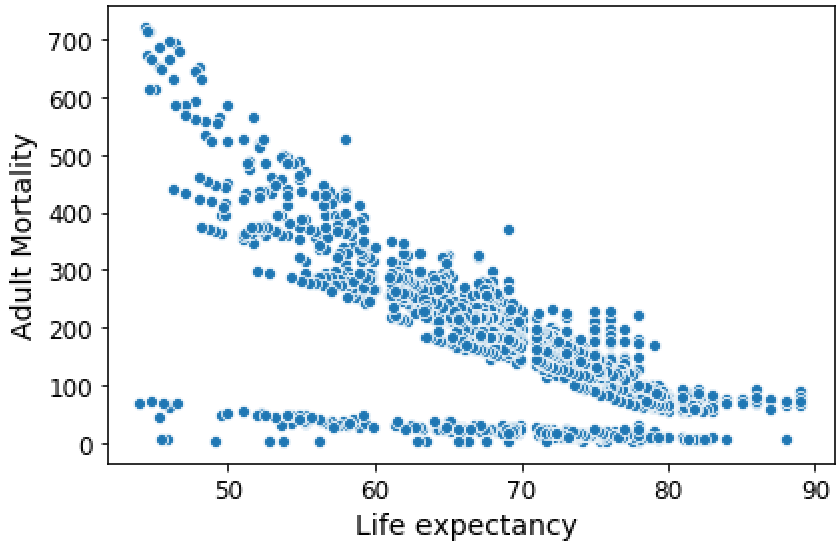


3、How infant and adult mortality affect their life expectancy?

In [57]:

```
sns.scatterplot(data=data,x='Life expectancy ',y='Adult Mortality')
save_fig("Adult")
```

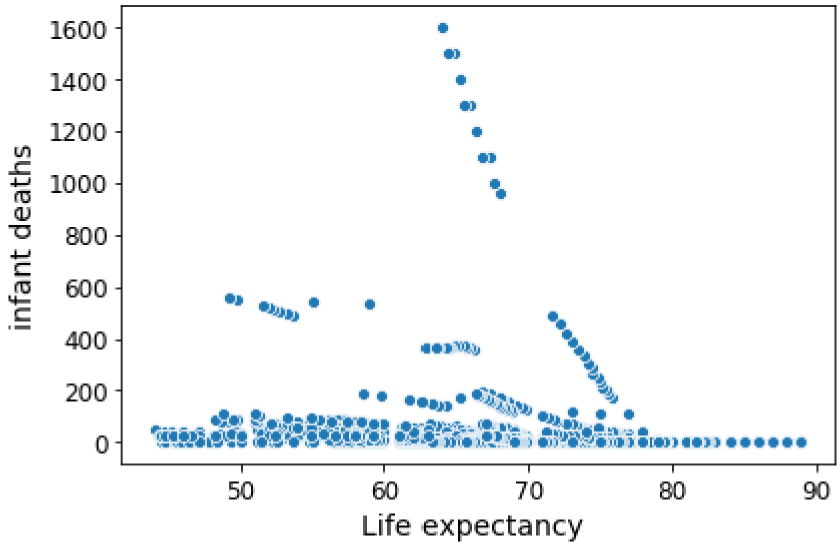
```
Saving figure Adult
```



In [58]:

```
sns.scatterplot(data=data,x='Life expectancy ',y='infant deaths')
save_fig("Infant")
```

```
Saving figure Infant
```



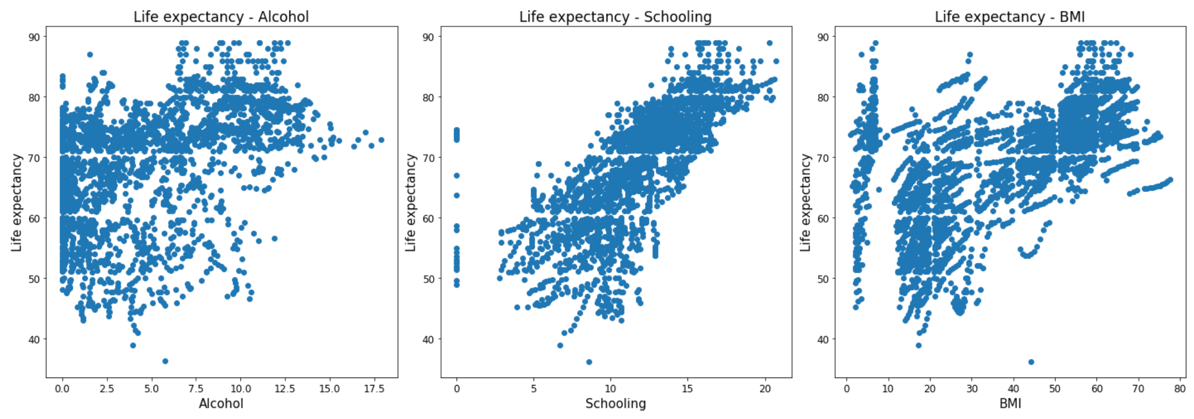
According to the graph, it can be seen directly that the higher the adult mortality rate, the shorter the life expectancy.

4&5&6、Is there a positive or negative correlation between life expectancy and eating habits, lifestyle, exercise, smoking, drinking, etc?

In [65]:

```
plt.figure(figsize=(20,7))
plt.subplot(1,3,1)
plt.scatter(df["Alcohol"], df["Life expectancy "])
plt.xlabel("Alcohol",fontsize=15)
plt.ylabel("Life expectancy",fontsize=15)
plt.title("Life expectancy - Alcohol",fontsize=17)
save_fig("Life expectancy - Alcohol")

plt.subplot(1,3,2)
plt.scatter(df["Schooling"], df["Life expectancy "])
plt.xlabel("Schooling",fontsize=15)
plt.ylabel("Life expectancy",fontsize=15)
plt.title("Life expectancy - Schooling",fontsize=17)
save_fig("Life expectancy - Schooling")

plt.subplot(1,3,3)
plt.scatter(df[" BMI "], df["Life expectancy "])
plt.xlabel("BMI",fontsize=15)
plt.ylabel("Life expectancy",fontsize=15)
plt.title("Life expectancy - BMI",fontsize=17)
save_fig("Life expectancy - BMI")
```

```
Saving figure Life expectancy - Alcohol
Saving figure Life expectancy - Schooling
Saving figure Life expectancy - BMI
```

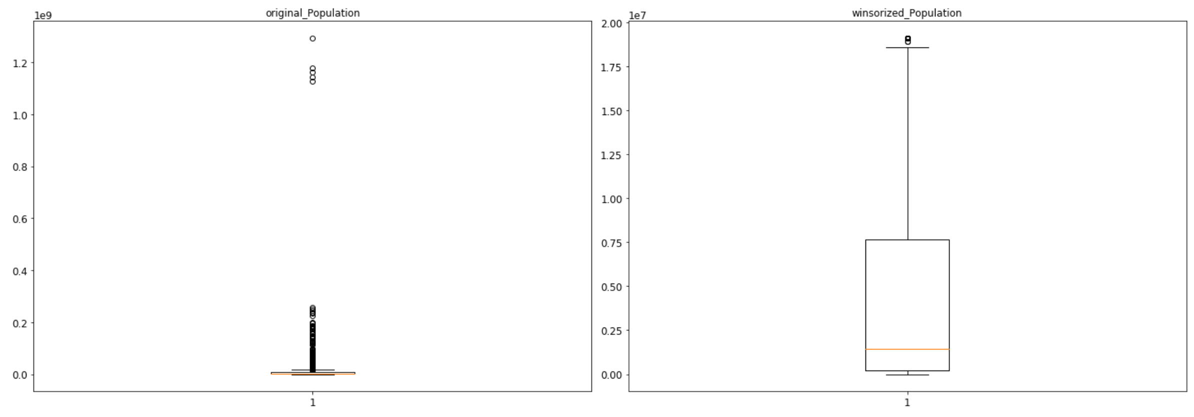


7、Do densely populated countries tend to reduce life expectancy?

In [71]:

```
# Winsorize Population
from scipy.stats.mstats import winsorize
plt.figure(figsize=(20,7))

plt.subplot(1,2,1)
original_Population = data['Population']
plt.boxplot(original_Population)
plt.title("original_Population")

plt.subplot(1,2,2)
winsorized_Population = winsorize(data['Population'],(0,0.14))
plt.boxplot(winsorized_Population)
plt.title("winsorized_Population")
save_fig("Saving figure Life expectancy - Population")
plt.show()
```

```
Saving figure Saving figure Life expectancy - Population
```

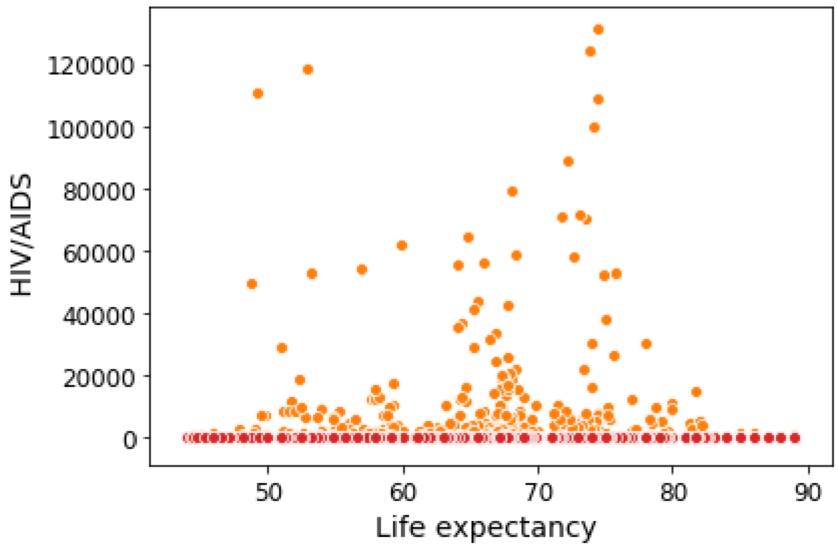


8、How does immunization coverage affect life expectancy?

In [73]:

```
for ill in ['Hepatitis B','Measles ','Polio',' HIV/AIDS']:
    sns.scatterplot(data=data,x='Life expectancy ',y=ill)
    save_fig("Life expectancy - {}".format(ill.replace("/","_")))
```

```
Saving figure Life expectancy - Hepatitis B
Saving figure Life expectancy - Measles
Saving figure Life expectancy - Polio
Saving figure Life expectancy -  HIV_AIDS
```



9、Build a model to predict life

```
Because it is to predict the life, it is not a single question of yes or no, so the decision tree
model cannot be used. In addition, it is related to time but time is not a consideration factor
for this question, so the time series model is not applicable, and the classification model of SVM
support vector machine is also Not applicable, the multiple linear regression model used in the
second question is more suitable in this case
```

In [146]:

```
X = data.loc[:,('Adult Mortality','infant deaths','Alcohol','percentage expenditure','Hepatitis B','
y = data.loc[:,'Life expectancy ']
X_train,X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2,random_state=100)
print('X_train.shape={}\ny_train.shape ={}\nX_test.shape={}\ny_test.shape={}'.format(X_train.shape,y
linreg = LinearRegression()
model = linreg.fit(X_train, y_train)
print(model)
# Model intercept after training
print(linreg.intercept_)
# Model weight after training (no change in the number of features)
print(linreg.coef_)
```

```
X_train.shape=(1319, 13)
y_train.shape =(1319,)
X_test.shape=(330, 13)
y_test.shape=(330,)
LinearRegression()
58.55340488252302
[-3.09202175e-02  9.04706992e-02  4.15897951e-02  6.72069509e-05
  2.53328031e-03 -2.31189687e-05 -7.05436858e-02  7.16205816e-03
  8.92022619e-02  2.01080598e-02  7.06860614e-05  2.02763442e-09
  1.97804299e+01]
```

In [147]:

```
feature_cols = ['Adult Mortality','infant deaths','Alcohol','percentage expenditure','Hepatitis B','
B = list(zip(feature_cols,linreg.coef_))
print(B)
```

```
[('Adult Mortality', -0.0309202174924528846), ('infant deaths', 0.09047069923204838),
('Alcohol', 0.04158979513910905б), ('percentage expenditure', 6.720695089256761e-0
5), ('Hepatitis B', 0.0025332803062751777), ('Measles ', -2.3118968727308963e-05),
('under-five deaths ', -0.07054368576047886), ('Polio', 0.007162058161366407), ('Tot
al expenditure', 0.08920226190175405), ('Diphtheria ', 0.020108059764838235), ('GD
P', 7.068606135318553e-05), ('Population', 2.027634421475355e-09), ('Income composit
ion of resources', 19.780429885954415)]
```

In [157]:

```
# predict
y_pred = linreg.predict(X_test)
print(y_pred)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('RMSE:',RMSE)
```

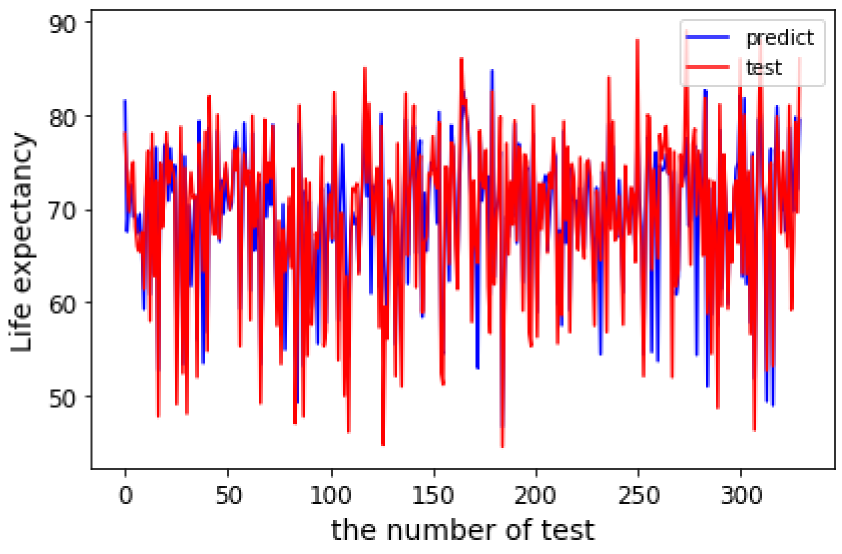
```
[81.45167005 67.55611667 70.68484742 72.55783106 69.15648683 69.41389694
 67.38411287 69.44726539 66.09615076 59.26243767 71.01706572 73.33219941
 60.81548355 76.59970507 63.04403691 76.5515882  52.70439639 72.9612413
 69.66869416 76.73452207 76.9484886  70.86810803 76.47108501 71.83220251
 74.6668418  56.64705915 66.63317918 75.86318074 53.38086378 75.58359781
 56.00680407 70.38743747 61.75191668 68.19878681 71.38806638 56.00837648
 79.32642614 70.20574868 53.50239318 76.17050579 56.80877681 78.7008715
 70.2744057  68.58141418 67.48516248 78.29592172 66.50307952 73.05374708
 69.4855109  74.30201903 70.94677951 69.89567515 70.58428241 75.94797336
 78.19620583 74.40589928 59.27116643 68.18482432 79.13283833 73.75250564
 73.53394264 61.2234892  78.0174199  65.5349457  67.64472929 72.93764977
 53.37976486 64.05344841 76.6612379  69.10786781 74.12866018 70.38235539
 78.92538666 66.29607619 59.62122293 66.44214836 56.52372636 70.48580301
 54.93622444 65.12133168 70.20419138 64.79732452 72.12773875 54.36469219
 49.34082179 79.04866225 70.42681652 53.19656486 71.95556776 65.61603236
 70.80047311 59.35669554 63.24946112 66.72433195 55.5797731  70.0313376
 72.15986085 67.5717035  57.77507892 72.63448841 70.52542635 66.47541214
 79.89333285 72.22705153 58.0491329  70.65809281 76.86321327 70.55651938
 62.86377184 54.27008297 67.53046001 69.62455429 68.36705278 69.03720854
 63.31152004 69.94206183 73.02134547 82.10623286 71.34921525 77.89732515
 60.95401666 72.48539935 71.23711267 73.27655114 61.6281873  80.12283567
 67.08193821 61.73628188 58.52409694 71.03960843 70.99517423 69.34348387
 55.49734818 68.21420945 61.80305737 64.98346735 70.64097692 79.51123615
 61.9810523  70.84471816 77.12387664 78.7665596  65.23027267 75.28141064
 77.31311858 58.4475599  71.69580663 65.48581498 71.80455842 74.93538139
 74.14497088 74.33197407 68.4333268  80.29887638 55.21235875 54.47670874
 70.13151482 72.50472792 62.27224899 78.86494493 76.01889096 71.10483076
 62.34650476 71.88514072 79.06637933 82.4999111  80.36774237 79.00768676
 74.14256844 69.63846639 70.84797133 63.5001359  52.96835301 78.08018407
 70.8451104  72.65194508 72.96958125 73.44537563 60.7978792  84.72373129
 62.64583966 69.72136871 71.30258304 76.10349087 46.70967926 62.3221795
 69.17918293 66.9318806  70.51509617 69.16731313 79.03670105 66.32117519
 76.96298814 73.83859605 62.12626645 73.36866739 74.44549429 59.55197539
 69.80510736 77.89578609 69.18775883 58.9464175  72.37927064 68.97911695
 72.41989226 73.56556457 65.56685833 68.98559165 71.83559745 76.22900489
 75.52535579 59.82650427 67.71612401 57.5395915  78.2643062  66.33151629
 73.44304148 59.15148433 74.42312674 70.87599723 70.87240194 65.94538213
 72.27307647 69.72948893 65.73205788 72.22633354 75.16292549 69.61534649
 66.70965119 62.15539086 72.26934541 68.15675742 54.43892302 74.02122641
 70.53927516 68.17146216 79.03072187 69.76705134 76.65962693 67.62473166
 66.9283216  73.07888941 67.40819236 61.04155385 74.42769087 70.44658773
 67.97536571 71.10772049 69.76603483 65.09526009 79.57110638 71.1476002
 68.0809494  54.42806798 68.85275124 77.58986881 77.24791669 54.68715852
 73.56899319 76.04373931 53.71653518 74.64305617 74.09473802 74.27299774
 75.80441016 73.79565654 74.00657106 67.24713864 74.01944091 60.82421215
 62.7125017  72.60033502 72.82087778 73.79244795 77.49615727 68.19807165
 68.79965031 74.68931847 76.75485409 54.37008492 75.86647342 71.90997635
 70.57133718 82.58383351 51.00790721 72.15126231 58.70767381 71.71773172
 68.55437466 57.50937249 78.37519047 61.4543271  74.71842866 69.56124276
 59.82355228 69.51603327 65.94349556 70.62728517 74.29565012 74.97986316
 82.07183729 62.70216965 81.77019138 61.9379359  73.3566646  57.83097873
 75.93462222 51.8380877  69.37250211 79.45649988 80.04689999 70.98250715
 67.5352621  49.41717667 71.13400332 76.41382338 48.98101752 73.7039444
```

```
  80.8848552  72.49588639 69.81837828 71.42513731 67.63195808 68.25554878
  78.64322457 73.40018724 69.73958108 79.73128634 72.06136426 79.4513647 ]
 RMSE: 4.415301829699224
```

In [158]:

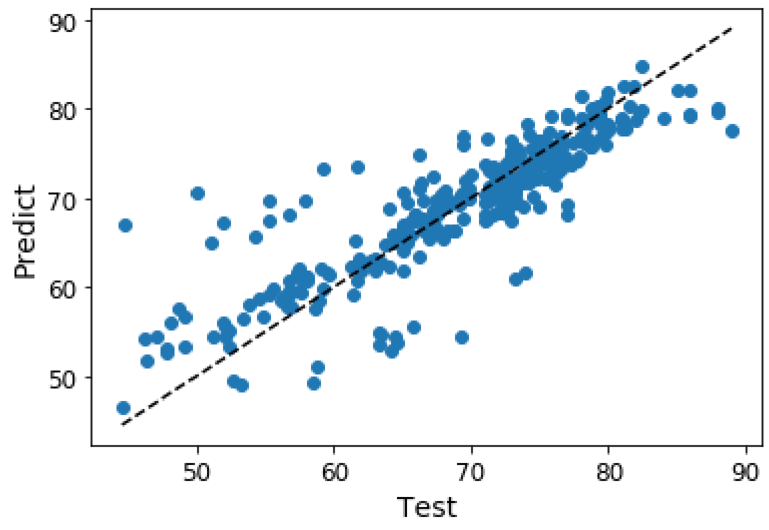
```
# Calculated error rate
# Make ROC curve
plt.figure()
plt.plot(range(len(y_pred)),y_pred,'b',label="predict")
plt.plot(range(len(y_pred)),y_test,'r',label="test")
# Show labels in the figure
plt.legend(loc="upper right")
plt.xlabel("the number of test")
plt.ylabel('Life expectancy')
save_fig("end_predict_plots")
plt.show()

plt.scatter(y_test, y_pred)
plt.plot([y_test.min(),y_test.max()], [y_test.min(),y_test.max()], 'k--')
plt.xlabel('Test')
plt.ylabel('Predict')
```

```
Saving figure end_predict_plots
```



Out[158]:

```
Text(0, 0.5, 'Predict')
```



In the figure, we can see that if the fit is complete, the scatter points should coincide with the straight line. Here, we find that there are more outliers in the place where y_test=90. The major disadvantage of the linear regression model is that it is very sensitive to outliers. , Will greatly affect the accuracy of the model, so in the next step, we will optimize the model based on this point

In [159]:

```
drop_index = data[data['Life expectancy ']==90].index.values
X = X.drop(drop_index)
y = y.drop(drop_index)
# The parameter random_state that divides the training set and the test set here are both 1, indicat
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=1)
# 对训练集进行训练
lr = LinearRegression()
lr.fit(X_train, y_train)
# 对测试集进行预测
y_pred = lr.predict(X_test)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('RMSE:',RMSE)
```

```
RMSE: 4.559245548886048
```

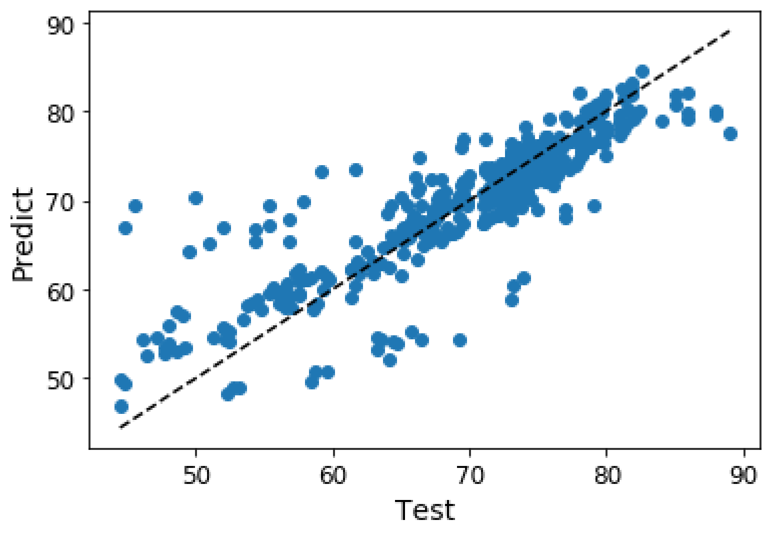
It can be seen that the RMSE of the model has become 4.559, which greatly improves the accuracy of the model. Here I am tuning the regression model mainly in two aspects: 1. Increase the number of training sets to improve training accuracy; 2. Remove outliers and improve accuracy

In [160]:

```
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(),y_test.max()], [y_test.min(),y_test.max()], 'k--')
plt.xlabel('Test')
plt.ylabel('Predict')
```

Out[160]:

```
Text(0, 0.5, 'Predict')
```



In [ ]: