作业 9 朱金秋 1220086621

17.7

给定 n 个样本的数据集 D，对它进行 bootstrap 抽样产生 n 个新的数据集 D'，由于是有放回的，有一部分样本会多次出现，在一次抽样中被抽到的概率为$\frac{1}{n}$，不被抽中的概率为 1-$\frac{1}{n}$。一共抽 n 次，当 n—>∞时，样本在 n 次抽样中被抽到概率。

$$\lim_{n\to\infty} 1-\left(1-\frac{1}{n}\right)^{n}=1-\frac{1}{e}\approx 0.632$$

7.18

a

Carry out a bootstrap analysis for the hormone data, like the one in Table 17.1, using B = 100 bootstrap samples.

In [1]:

```
import pandas as pd
import numpy as np
hormone = pd.read_csv('hormone.csv')
df = hormone.loc[:,['Lot','hrs','amount']]
data = np.array(df)
dummies = pd.get_dummies(
    df,
    columns=['Lot'],
    prefix=['Lot'],
    prefix_sep="_",
    dummy_na=False,
    drop_first=False
)

dummies['Lot'] = df['Lot']  #添加虚拟变量

X = dummies.loc[:,['Lot_A','Lot_B','Lot_C','hrs']]
y = dummies.loc[:,'amount']
```

In [58]:

```
model = linear_model.LinearRegression().fit(X,y)  #对原始数据建立线性回归模型并预测
X_predict = model.predict(X)
err_X_F = np.mean((X_predict - y)**2)  #  err(x,F^)
err_X_F
```

Out[58]:

2.1952182295633165

In [4]:

```
data = np.array(dummies)
```

In [6]:

```
from sklearn import linear_model
```

In [301]:

```
def get_out_of_bag(sample_i):
    dummies_list = dummies.values.tolist()
    for i in dummies.values.tolist():
        if i in pd.DataFrame(sample_i).values.tolist():
              dummies_list.remove(i)
    X_out_of_bag = dummies_list
    return(X_out_of_bag)
```

In [43]:

```
N = 100  #100次抽样
k = len(data)
data_Bs = []
error_star_Fhead = []
error_star_fhead_star = []
error_0 = []
for i in range(N):
    sample_i = []
    for j in np.random.choice(range(k),k):
        sample = list(data[j,:])
        sample_i.append(sample)
    X_sample = pd.DataFrame(sample_i).iloc[:,[2,3,4,0]]
    y_sample = pd.DataFrame(sample_i).iloc[:,[1]]
    model =linear_model.LinearRegression().fit(X_sample,y_sample) #建立线性回归模型
    y_predict_X = model.predict(X)
    y_predict_X_sample = model.predict(X_sample)
    error_star_Fhead_i = np.mean((y_predict_X.reshape(k,)-y)**2)
    error_star_fhead_star_i = np.mean((np.array(y_sample).reshape(k,) -y_predict_X_sample.reshape(k,
    error_star_Fhead.append(error_star_Fhead_i)
    error_star_fhead_star.append(error_star_fhead_star_i)

    #out of bags
    a = get_out_of_bag(sample_i)
    X_out_of_bag = pd.DataFrame(a).iloc[:,[2,3,4,0]]
    y_out_of_bag = pd.DataFrame(a).iloc[:,1]
    k_out_of_bag = X_out_of_bag.shape[0]
    y_out_of_bag_predict = model.predict(X_out_of_bag)
    error_0_i = np.mean(((y_out_of_bag_predict.reshape(k_out_of_bag,) -y_out_of_bag)**2))
    error_0.append(error_0_i)  #error_0
```

In [8]:

```
a = pd.DataFrame(error_star_Fhead,columns = ['error(x*,F^)'])
b = pd.DataFrame(error_star_fhead_star,columns = ['error(x*,F*^)'])
df_1 = pd.concat([a,b],axis = 1)
df_1['error(x*,F^) - error(x*,F*^)'] = df_1['error(x*,F^)'] - df_1['error(x*,F*^)']
df_1,df_1['error(x*,F^)'].mean(),df_1['error(x*,F*^)'].mean(),df_1['error(x*,F^) - error(x*,F*^)'].m
```

Out[8]:

```
(    error(x*,F^)  error(x*,F*^)  error(x*,F^) - error(x*,F*^)
 0       2.594250       1.903138                     0.691112
 1       3.859426       1.353592                     2.505835
 2       2.382012       3.039969                    -0.657957
 3       2.522658       1.293612                     1.229046
 4       2.593062       1.551758                     1.041304
 ..           ...            ...                          ...
 95      2.274997       2.782179                    -0.507182
 96      2.653488       1.136391                     1.517097
 97      2.440890       2.635267                    -0.194377
 98      2.452817       3.092355                    -0.639538
 99      2.449721       2.142901                     0.306820

 [100 rows x 3 columns],
 2.647339207136122,
 1.906708056767622,
 0.7406311503685009)
```

通过实验结果可以得出100次Bootstrap 抽样的error(x,*F^) error(x*,F^) *error(x*,F^) - error(x,*F^)，其结果与书上结果接近。

根据书上公式计算得到efsino_0

In [161]:

```
efsino_0 = np.mean(error_0)
efsino_0
```

Out[161]:

3.6686082512787346

计算得到efsino_0如上

In [318]:

```
err_632 = 0.368*err_X_F  + 0.632*efsino_0#compute the .632 estimator for these data
err_632
```

Out[318]:

3.034808447844411

b

Calculate the average prediction error Ej for observa-tions that appear exactly j times in the bootstrap sam-ple used for their prediction, for j = 0, 1, 2, ... Graph Ej against j and give an explanation for the results.

In [319]:

```
def get_df_times(sample_i):  #得到各样本的抽样次数
    count_i = []
    for m in sample_i:
        count = 0
        for n in sample_i:
            if m==n:
                count+=1
        count_i.append(count)
    df_times = pd.concat([pd.DataFrame(sample_i),pd.DataFrame(count_i)],axis = 1)
    return df_times
```

In [297]:

```
N = 100  #100次抽样
k = len(data)
data_Bs = []
error_0 = []
Ej = []
for i in range(N):
    sample_i = []
    for j in np.random.choice(range(k),k):
        sample = list(data[j,:])
        sample_i.append(sample)
    X_sample = pd.DataFrame(sample_i).iloc[:,[2,3,4,0]]
    y_sample = pd.DataFrame(sample_i).iloc[:,[1]]
    model =linear_model.LinearRegression().fit(X_sample,y_sample) #建立线性回归模型
    #out of bags E0
    a = get_out_of_bag(sample_i)
    X_out_of_bag = pd.DataFrame(a).iloc[:,[2,3,4,0]]
    y_out_of_bag = pd.DataFrame(a).iloc[:,1]
    k_out_of_bag = X_out_of_bag.shape[0]
    y_out_of_bag_predict = model.predict(X_out_of_bag)
    error_0_i = np.mean(((y_out_of_bag_predict.reshape(k_out_of_bag,) -y_out_of_bag)**2))
    error_0.append(error_0_i)


# efsino_0 = np.mean(error_0)#E0
    #Ej 出现过j次的样本的平均误差
    sample_times = get_df_times(sample_i)
    error_j = []
    for m in range(1,sample_times.iloc[:,-1].max()+1):
        train = sample_times[sample_times.iloc[:,-1]==m]
        if train.shape[0] == 0:
            error_j_m = [None]  #如果不存在则为0
        else:
            X_times = train.iloc[:,[2,3,4,0]]
            y_times =train.iloc[:,1]
            k_times = X_times.shape[0]
            y_times_predict = model.predict(X_times)
            error_j_m = np.mean(((y_times_predict.reshape(k_times,) -y_times)**2))
        error_j.append(error_j_m)
    Ej.append(error_j)
```

In [298]:

```
a = pd.DataFrame(Ej)
for p in range(pd.DataFrame(Ej).shape[0]):
    for q in range(pd.DataFrame(Ej).shape[1]):
        if pd.DataFrame(Ej).iloc[p,q] == [None]:
            a.iloc[p,q] = None

df_Ej = pd.concat([pd.DataFrame(error_0),a],axis = 1)
#df_Ej
columns = []
for q in range(df_Ej.shape[1]):
    names = 'E' +str(q)+'_times'
    columns.append(names)
df_Ej.columns = columns
df_Ej  # error Ej for observa-tions
```

	E0_times	E1_times	E2_times	E3_times	E4_times	E5_times	E6_times
0	3.568603	2.958571	2.054310	0.214168	None	None	0.39345
1	10.434633	1.400653	1.052633	0.682098	1.356756	None	NaN
2	3.273085	2.055680	1.386965	0.125747	0.177171	None	NaN
3	4.066045	2.929054	1.570950	0.276814	None	None	NaN
4	3.586765	2.308593	1.908030	0.264197	3.663821	None	NaN
...	...	...	...	...	...	...	...
95	1.383269	3.367115	1.537844	2.866085	None	None	NaN
96	4.481933	2.250578	1.403823	None	None	None	NaN
97	2.167399	2.602242	2.040581	0.519136	None	None	NaN
98	2.584309	2.547515	1.999425	1.783992	None	None	NaN
99	5.847033	1.974293	0.966222	1.742108	None	None	NaN

100次bootstarp后，prediction error Ej for observa-tions that appear exactly j times in the bootstrap sample used for their prediction如上，可以看出有的样本被抽中了4、5、6次，但有的样本没被抽中这么多次，但我还是把他们放进来统计其估计误差，多次bootstrap实验后，可以发现，大多数样本能 达到0,1,2次有抽样结果

他们平均误差如下。样本被抽中1-6次的平均误差如下

In [310]:

```
list(df_Ej.mean())  #
```

Out[310]:

```
[3.6309706548954943,
 2.509854675794668,
 1.6508201547920591,
 1.2446181831610297,
 1.3722203602511598,
 0.8899533839730376,
 0.2694859562751226]
```

绘图

In [312]:

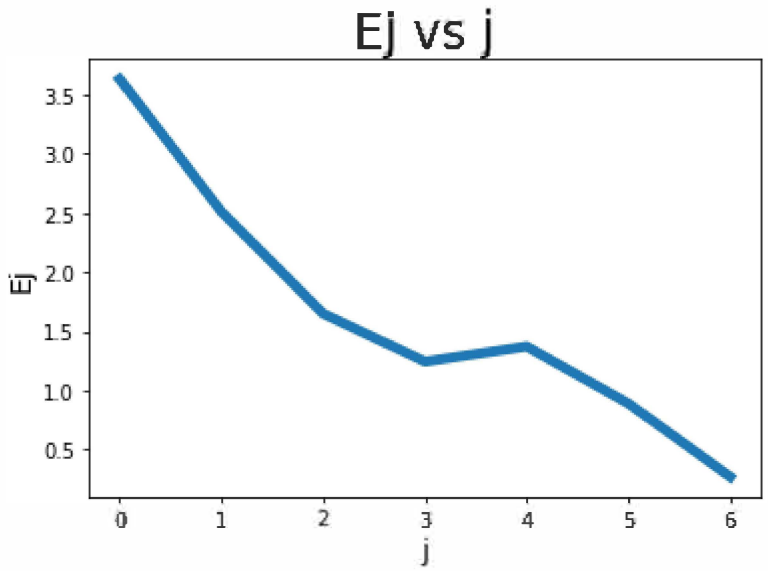
```
import matplotlib.pyplot as plt

x_l = list(range(len(df_Ej.mean())))
y_l = list(df_Ej.mean())
```

In [314]:

```
import matplotlib.pyplot as plt
plt.plot(x_l,y_l,linewidth=5)
#plt.plot(squares)
 #设置图表标题，并给坐标轴加标签
plt.title("Ej vs j",fontsize=24)
plt.xlabel("j",fontsize=14)
plt.ylabel("Ej",fontsize=14)
```

Out[314]:

Text(0, 0.5, 'Ej')



随着J的增加，平均预测误差有明显的下降趋势

In [ ]: