hw7_answer

February 26, 2019

1 Assignment 7

1.0.1 MACS 30150, Dr. Evans

1.0.2 Dongcheng Yang

problem1 (a)

In [5]:
```
import pandas as pd
df = pd.read_csv('data/strongdrink.txt')
df.head()
```

Out[5]:

	cultivar	alco	malic	ash	alk	magn	tot_phen	flav	nonfl_phen	proanth	color_int	hue	OD280rat	proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

In [6]:
```
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegression
import numpy as np

X = df[['alco', 'malic', 'tot_phen', 'color_int']]
y = df["cultivar"]

X_train, X_test, y_train, y_test = \
  train_test_split(X, y, test_size = 0.25,random_state=20)

clf = LogisticRegression(solver='lbfgs', \
        multi_class='multinomial').fit(X_train, y_train)
```

```
        pd.DataFrame({"1":np.append(clf.intercept_[0],clf.coef_[0]),
                      "2":np.append(clf.intercept_[1],clf.coef_[1])},
                index=["beta_0","beta_1",'beta_2','beta_3','beta_4'])
```

Out[6]:

	1	2
beta_0	-24.027617	22.780733
beta_1	1.701734	-1.466297
beta_2	-0.265788	-0.332951
beta_3	1.224101	0.663556
beta_4	0.022507	-0.922682

```
In [8]: from sklearn.metrics import classification_report
        y_pred = clf.predict(X_test)
        print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
1	0.87	1.00	0.93	13
2	1.00	0.90	0.95	21
3	1.00	1.00	1.00	10
avg / total	0.96	0.95	0.96	44

```
In [9]: df["cultivar"].value_counts()
```

```
Out[9]: 2    71
        1    59
        3    46
        Name: cultivar, dtype: int64
```

The error rates are 13%, 0%, and 0% for group 1,2, and 3, respectively.
From the classification report, we can see that the model is best at predicting the third group with the highest f1-score. And the third group is not the one with the most observations.

```
In [10]: mean_squared_err = np.mean((y_test != y_pred) ** 2)
         print("Mean squared error is {}".format(mean_squared_err))
```

```
Mean squared error is 0.045454545454545456
```

(b)

```
In [11]: from sklearn.model_selection import LeaveOneOut
         Xvars = df[['alco', 'malic', 'tot_phen', 'color_int']].values
         yvars = df['cultivar'].values
         N_loo = Xvars.shape[0]
         loo = LeaveOneOut()
```

```
loo.get_n_splits(Xvars)
MSE_vec = np.zeros(N_loo)

ytest = np.zeros(N_loo)
ypred = np.zeros(N_loo)

for train_index, test_index in loo.split(Xvars):
    X_train, X_test = Xvars[train_index], Xvars[test_index]
    y_train, y_test = yvars[train_index], yvars[test_index]
    ytest[test_index] = y_test
    LogReg = LogisticRegression()
    LogReg.fit(X_train, y_train)
    y_pred = LogReg.predict(X_test)
    ypred[test_index] = y_pred
    if y_test == y_pred:
        MSE_vec[test_index] = 0
    else:
        MSE_vec[test_index] = 1
    print('MSE for test set', test_index, ' is', MSE_vec[test_index])
```

```
MSE for test set [0]  is [0.]
MSE for test set [1]  is [0.]
MSE for test set [2]  is [0.]
MSE for test set [3]  is [0.]
MSE for test set [4]  is [0.]
MSE for test set [5]  is [0.]
MSE for test set [6]  is [0.]
MSE for test set [7]  is [0.]
MSE for test set [8]  is [0.]
MSE for test set [9]  is [0.]
MSE for test set [10]  is [0.]
MSE for test set [11]  is [1.]
MSE for test set [12]  is [0.]
MSE for test set [13]  is [0.]
MSE for test set [14]  is [0.]
MSE for test set [15]  is [0.]
MSE for test set [16]  is [0.]
MSE for test set [17]  is [0.]
MSE for test set [18]  is [0.]
MSE for test set [19]  is [0.]
MSE for test set [20]  is [0.]
MSE for test set [21]  is [1.]
MSE for test set [22]  is [1.]
MSE for test set [23]  is [1.]
MSE for test set [24]  is [1.]
MSE for test set [25]  is [1.]
MSE for test set [26]  is [0.]
MSE for test set [27]  is [1.]
```

```
MSE for test set [28]  is [0.]
MSE for test set [29]  is [0.]
MSE for test set [30]  is [0.]
MSE for test set [31]  is [0.]
MSE for test set [32]  is [1.]
MSE for test set [33]  is [0.]
MSE for test set [34]  is [1.]
MSE for test set [35]  is [0.]
MSE for test set [36]  is [0.]
MSE for test set [37]  is [1.]
MSE for test set [38]  is [1.]
MSE for test set [39]  is [0.]
MSE for test set [40]  is [0.]
MSE for test set [41]  is [1.]
MSE for test set [42]  is [0.]
MSE for test set [43]  is [1.]
MSE for test set [44]  is [0.]
MSE for test set [45]  is [0.]
MSE for test set [46]  is [0.]
MSE for test set [47]  is [0.]
MSE for test set [48]  is [0.]
MSE for test set [49]  is [0.]
MSE for test set [50]  is [0.]
MSE for test set [51]  is [0.]
MSE for test set [52]  is [0.]
MSE for test set [53]  is [0.]
MSE for test set [54]  is [0.]
MSE for test set [55]  is [0.]
MSE for test set [56]  is [0.]
MSE for test set [57]  is [0.]
MSE for test set [58]  is [0.]
MSE for test set [59]  is [0.]
MSE for test set [60]  is [0.]
MSE for test set [61]  is [1.]
MSE for test set [62]  is [0.]
MSE for test set [63]  is [1.]
MSE for test set [64]  is [0.]
MSE for test set [65]  is [1.]
MSE for test set [66]  is [1.]
MSE for test set [67]  is [1.]
MSE for test set [68]  is [0.]
MSE for test set [69]  is [0.]
MSE for test set [70]  is [0.]
MSE for test set [71]  is [0.]
MSE for test set [72]  is [0.]
MSE for test set [73]  is [0.]
MSE for test set [74]  is [0.]
MSE for test set [75]  is [0.]
```

```
MSE for test set [76]  is [0.]
MSE for test set [77]  is [0.]
MSE for test set [78]  is [0.]
MSE for test set [79]  is [0.]
MSE for test set [80]  is [0.]
MSE for test set [81]  is [0.]
MSE for test set [82]  is [0.]
MSE for test set [83]  is [1.]
MSE for test set [84]  is [0.]
MSE for test set [85]  is [0.]
MSE for test set [86]  is [0.]
MSE for test set [87]  is [0.]
MSE for test set [88]  is [0.]
MSE for test set [89]  is [0.]
MSE for test set [90]  is [0.]
MSE for test set [91]  is [0.]
MSE for test set [92]  is [0.]
MSE for test set [93]  is [0.]
MSE for test set [94]  is [0.]
MSE for test set [95]  is [0.]
MSE for test set [96]  is [0.]
MSE for test set [97]  is [0.]
MSE for test set [98]  is [1.]
MSE for test set [99]  is [0.]
MSE for test set [100]  is [0.]
MSE for test set [101]  is [0.]
MSE for test set [102]  is [0.]
MSE for test set [103]  is [0.]
MSE for test set [104]  is [0.]
MSE for test set [105]  is [0.]
MSE for test set [106]  is [0.]
MSE for test set [107]  is [0.]
MSE for test set [108]  is [0.]
MSE for test set [109]  is [0.]
MSE for test set [110]  is [0.]
MSE for test set [111]  is [0.]
MSE for test set [112]  is [0.]
MSE for test set [113]  is [0.]
MSE for test set [114]  is [0.]
MSE for test set [115]  is [0.]
MSE for test set [116]  is [0.]
MSE for test set [117]  is [0.]
MSE for test set [118]  is [0.]
MSE for test set [119]  is [0.]
MSE for test set [120]  is [0.]
MSE for test set [121]  is [1.]
MSE for test set [122]  is [0.]
MSE for test set [123]  is [0.]
```

```
MSE for test set [124]  is [0.]
MSE for test set [125]  is [0.]
MSE for test set [126]  is [0.]
MSE for test set [127]  is [0.]
MSE for test set [128]  is [0.]
MSE for test set [129]  is [0.]
MSE for test set [130]  is [1.]
MSE for test set [131]  is [0.]
MSE for test set [132]  is [0.]
MSE for test set [133]  is [0.]
MSE for test set [134]  is [1.]
MSE for test set [135]  is [0.]
MSE for test set [136]  is [0.]
MSE for test set [137]  is [0.]
MSE for test set [138]  is [1.]
MSE for test set [139]  is [0.]
MSE for test set [140]  is [0.]
MSE for test set [141]  is [0.]
MSE for test set [142]  is [0.]
MSE for test set [143]  is [0.]
MSE for test set [144]  is [0.]
MSE for test set [145]  is [0.]
MSE for test set [146]  is [0.]
MSE for test set [147]  is [0.]
MSE for test set [148]  is [0.]
MSE for test set [149]  is [0.]
MSE for test set [150]  is [0.]
MSE for test set [151]  is [0.]
MSE for test set [152]  is [0.]
MSE for test set [153]  is [0.]
MSE for test set [154]  is [0.]
MSE for test set [155]  is [0.]
MSE for test set [156]  is [0.]
MSE for test set [157]  is [0.]
MSE for test set [158]  is [0.]
MSE for test set [159]  is [0.]
MSE for test set [160]  is [0.]
MSE for test set [161]  is [0.]
MSE for test set [162]  is [0.]
MSE for test set [163]  is [0.]
MSE for test set [164]  is [0.]
MSE for test set [165]  is [0.]
MSE for test set [166]  is [0.]
MSE for test set [167]  is [0.]
MSE for test set [168]  is [0.]
MSE for test set [169]  is [0.]
MSE for test set [170]  is [0.]
MSE for test set [171]  is [0.]
```

```
MSE for test set [172]  is [0.]
MSE for test set [173]  is [0.]
MSE for test set [174]  is [0.]
MSE for test set [175]  is [0.]
```

```
In [12]: MSE_loo = MSE_vec.mean()
         MSE_loo_std = MSE_vec.std()
         print('test estimate MSE loocv=', MSE_loo,
               '\ntest estimate MSE standard err=', MSE_loo_std)
```

```
test estimate MSE loocv= 0.13636363636363635
test estimate MSE standard err= 0.3431742925123068
```

```
In [13]: print(classification_report(ytest, ypred))
```

```
             precision    recall  f1-score   support

        1.0       0.84      0.78      0.81        59
        2.0       0.83      0.89      0.86        71
        3.0       0.96      0.93      0.95        46

avg / total       0.86      0.86      0.86       176
```

The error rates are 16%, 17%, and 4% for group 1,2, and 3, respectively. The error rates increase as compared with part (a) for all of the three groups.

(c)

```
In [14]: from sklearn.model_selection import KFold
         k = 4
         kf = KFold(k, random_state=10, shuffle=True)
         kf.get_n_splits(Xvars)

         MSE_vec_kf = np.zeros(k)

         ytest = np.zeros(N_loo)
         ypred = np.zeros(N_loo)

         k_ind = int(0)
         for train_index, test_index in kf.split(Xvars):
             X_train, X_test = Xvars[train_index], Xvars[test_index]
             y_train, y_test = yvars[train_index], yvars[test_index]
             ytest[test_index] = y_test

             LogReg = LogisticRegression()
```

```
            LogReg.fit(X_train, y_train)
            y_pred = LogReg.predict(X_test)
            ypred[test_index] = y_pred

            n = len(y_pred)
            err=[1 if y_test[i] != y_pred[i] else 0 for i in range(n)]
            MSE_vec_kf[k_ind]=np.mean(err)
            print('MSE for test set', k_ind, ' is', MSE_vec_kf[k_ind])
            k_ind += 1
```

```
MSE for test set 0  is 0.22727272727272727
MSE for test set 1  is 0.22727272727272727
MSE for test set 2  is 0.13636363636363635
MSE for test set 3  is 0.09090909090909091
```

```
In [15]: MSE_kf = MSE_vec_kf.mean()
         MSE_kf_std = MSE_vec_kf.std()
         print('test estimate MSE k-fold=', MSE_kf,
               '\ntest estimate MSE standard err=', MSE_kf_std)
```

```
test estimate MSE k-fold= 0.17045454545454544
test estimate MSE standard err= 0.05904718662166627
```

```
In [16]: print(classification_report(ytest, ypred))
```

```
             precision    recall  f1-score   support

        1.0       0.78      0.73      0.75        59
        2.0       0.79      0.85      0.82        71
        3.0       0.96      0.93      0.95        46

avg / total       0.83      0.83      0.83       176
```

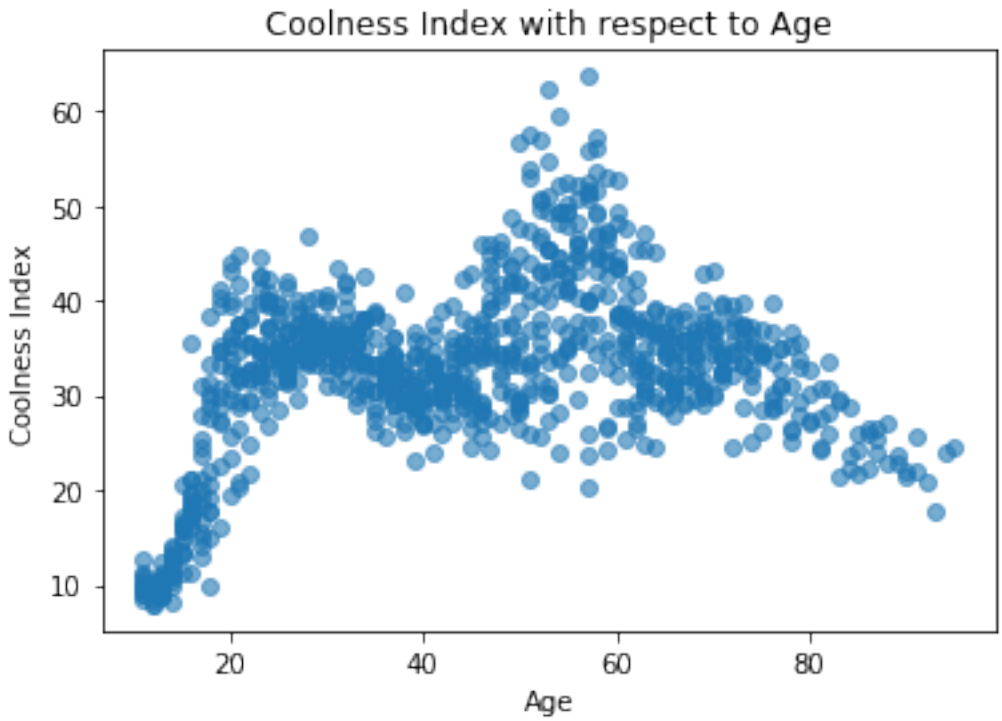
The error rates are 22%, 21%, and 4% for group 1,2, and 3, respectively. The error rates increase as compared with part (b) for group 1 and 2. The error rate for group 3 does not change compared to part (b), but still larger than part (a).

problem2 (a)

```
In [17]: df2=pd.read_csv("data/CoolIndex.txt",names=["Age","Cool"])
         df2.head()
```

```
Out[17]:     Age       Cool
         0  11.0  10.981602
         1  11.0  11.364925
         2  11.0  10.190227
         3  11.0   9.903725
         4  11.0   8.997918
```

```
In [19]: import matplotlib.pyplot as plt
         plt.scatter(df2['Age'],df2['Cool'],alpha=0.6)
         plt.xlabel("Age")
         plt.ylabel("Coolness Index")
         plt.title("Coolness Index with respect to Age")
         plt.show()
```



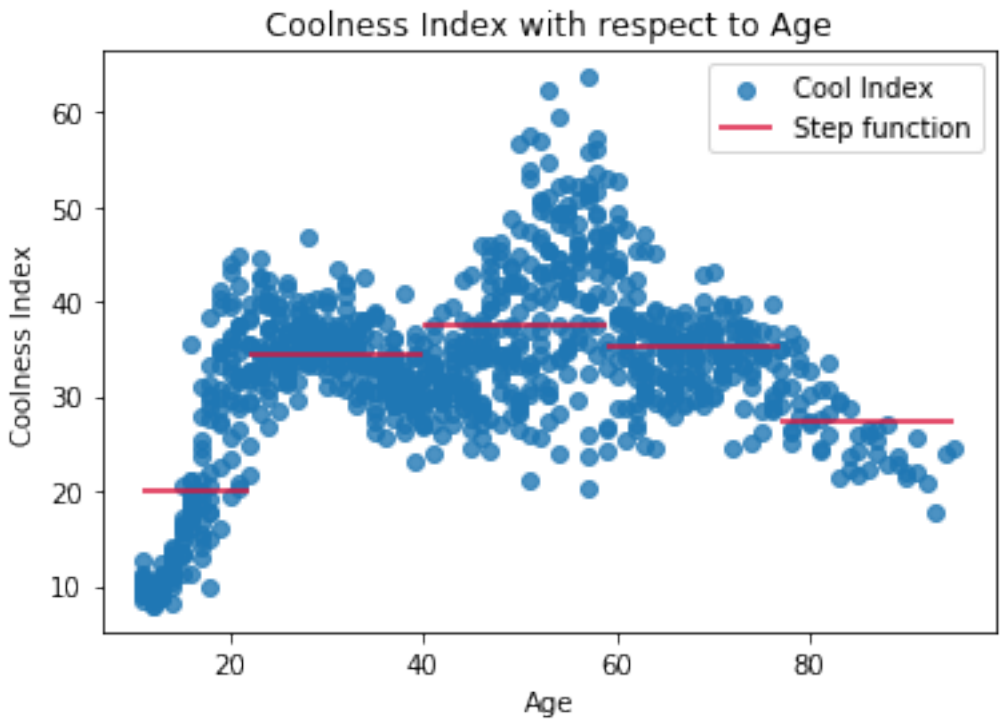
(b)

```
In [20]: import statsmodels.api as sm

         df2["g1"]=np.where((df2.Age>=11) & (df2.Age<22),1,0)
         df2["g2"]=np.where((df2.Age>=22) & (df2.Age<40),1,0)
         df2["g3"]=np.where((df2.Age>=40) & (df2.Age<59),1,0)
         df2["g4"]=np.where((df2.Age>=59) & (df2.Age<77),1,0)
         df2["g5"]=np.where((df2.Age>=77) & (df2.Age<=95),1,0)

         X=df2[["g1","g2","g3","g4","g5"]]
         res=sm.OLS(df2.Cool,X).fit()
         print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Cool   R-squared:                       0.429
```

```
Model:                            OLS   Adj. R-squared:                  0.427
Method:                 Least Squares   F-statistic:                     178.7
Date:                Tue, 26 Feb 2019   Prob (F-statistic):          3.73e-114
Time:                        23:53:46   Log-Likelihood:                -3214.5
No. Observations:                 956   AIC:                             6439.
Df Residuals:                     951   BIC:                             6463.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
g1            20.1025      0.562     35.746      0.000      18.999      21.206
g2            34.4758      0.431     80.006      0.000      33.630      35.321
g3            37.6351      0.424     88.814      0.000      36.804      38.467
g4            35.2254      0.485     72.560      0.000      34.273      36.178
g5            27.2964      0.936     29.175      0.000      25.460      29.132
==============================================================================
Omnibus:                       80.102   Durbin-Watson:                   1.236
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              101.718
Skew:                           0.714   Prob(JB):                     8.17e-23
Kurtosis:                       3.719   Cond. No.                         2.21
==============================================================================

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

```
In [21]: values = [20.1025, 34.4758, 37.6351, 35.2254, 27.2964]
         plt.scatter(df2['Age'],df2['Cool'], alpha=0.8,label='Cool Index')
         x_min = np.array([11, 22, 40, 59, 77])
         x_max = np.array([22, 40, 59, 77, 95])
         plt.hlines(values, x_min, x_max, color='crimson', label='Step function')
         plt.xlabel("Age")
         plt.ylabel("Coolness Index")
         plt.title("Coolness Index with respect to Age")
         plt.legend()
         plt.show()
```



From the OLS regression results above, the estimated step function values for each bin are 20.1025, 34.4758, 37.6351, 35.2254, 27.2964 respectively.

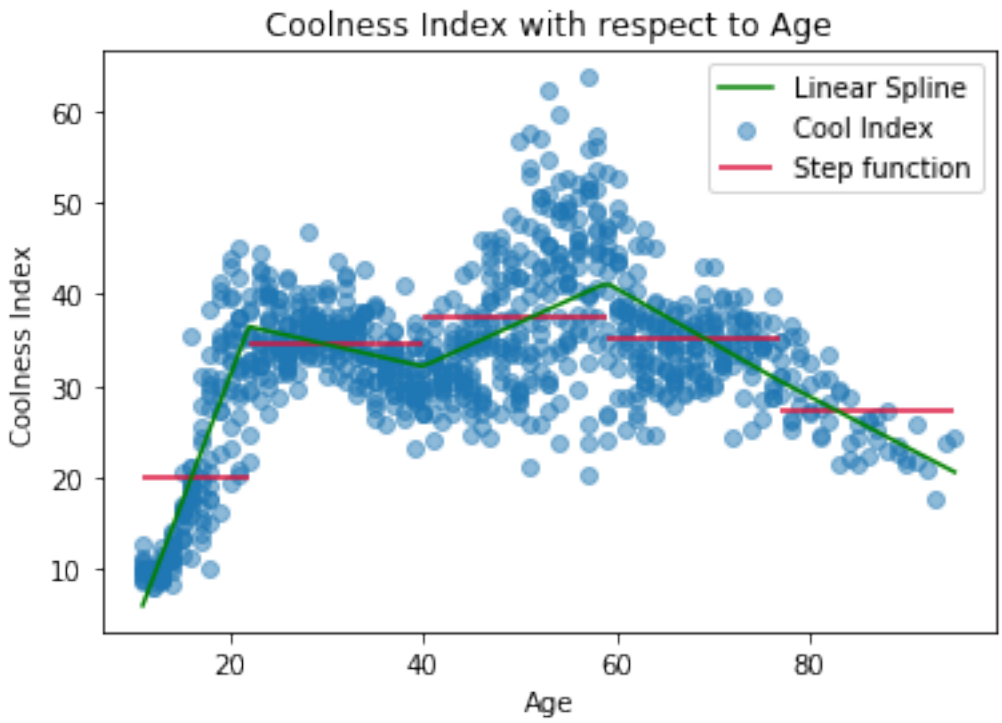
```
In [22]: res.predict([0,0,0,1,0])
```

```
Out[22]: array([35.22540004])
```

The predicted coolness of a 73-year old from the stepwise function is 35.2254.

(c)

```
In [23]: from scipy.interpolate import LSQUnivariateSpline
         t = np.array([22,40,59,77])
         df2.sort_index(0, ascending=True, inplace=True)
         group = df2.groupby('Age', as_index = False).mean()
         usl = LSQUnivariateSpline(group.Age.values, group.Cool.values, t, k=1)
         age2 = np.linspace(11,95,100)

         plt.scatter(df2['Age'],df2['Cool'], alpha=0.5,label='Cool Index')
         plt.hlines(values, x_min, x_max, color='crimson', label='Step function')
         plt.plot(age2, usl(age2), color='green', label='Linear Spline')
         plt.xlabel("Age")
         plt.ylabel("Coolness Index")
         plt.title("Coolness Index with respect to Age")
         plt.legend()
         plt.show()
```

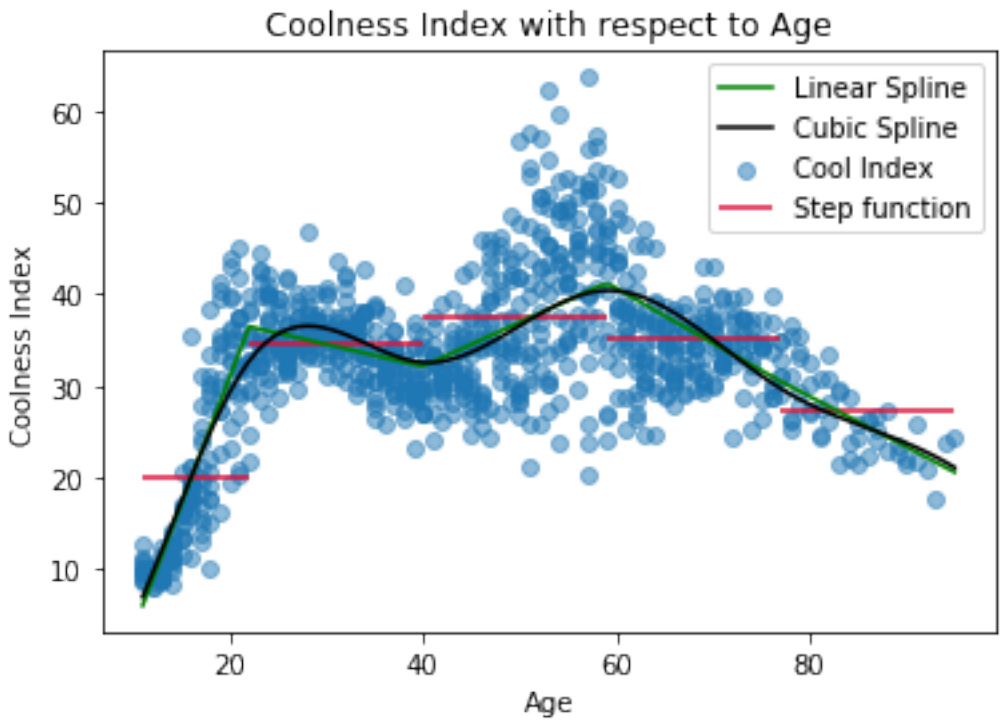



```
In [24]: print('The predicted coolness index of a 73-year-old \
         \nperson from the linear spline is', usl(73))
```

```
The predicted coolness index of a 73-year-old
person from the linear spline is 32.86784862349653
```

(d)

```
In [25]: usl2 = LSQUnivariateSpline(group.Age.values, group.Cool.values, t, k=3)

         plt.scatter(df2['Age'],df2['Cool'], alpha=0.5,label='Cool Index')
         plt.hlines(values, x_min, x_max, color='crimson', label='Step function')
         plt.plot(age2, usl(age2), color='green', label='Linear Spline')
         plt.plot(age2, usl2(age2), 'k-', label='Cubic Spline')
         plt.xlabel("Age")
         plt.ylabel("Coolness Index")
         plt.title("Coolness Index with respect to Age")
         plt.legend()
         plt.show()
```


Coolness Index with respect to Age
Linear Spline
Cubic Spline
Cool Index
Step function
Coolness Index
Age
10
20
30
40
50
60
20
40
60
80