```
 1 import pandas as pd
 2 import numpy as np
 3 from sklearn.linear_model import LogisticRegression
 4 from matplotlib import pyplot as plt
 5 import sklearn.linear_model as linear_model
 6 from sklearn.metrics import accuracy_score
 7 from sklearn import preprocessing
 8 from sklearn.model_selection import train_test_split
 9 from sklearn.model_selection import KFold
10 from sklearn import metrics
11
12 df = pd.read_csv('ML_HW_Data_CellDNA.csv', header =
   None)
13 df.head()
14
15 # Binarizing the data
16 new_cellDna = []
17 for i in df[13]:
18     if i == 0:
19         new_cellDna.append(i)
20     else:
21         i = 1
22         new_cellDna.append(i)
23
24 df["dependant_variable"] = new_cellDna
25
26 # removing the redundant column
27
28 df.drop([7, 8, 13], axis=1, inplace=True)
29 df = df.rename(columns={9: 7, 10: 8, 11: 9, 12: 10})
30 print(df)
31 df.info()
32
33 # standardizing
34
35 X_need_scaling = df.drop(labels=['dependant_variable'
   ], axis=1)
36 X = preprocessing.scale(X_need_scaling)
37 y = df["dependant_variable"]
38
39 X_train, X_test, y_train, y_test = train_test_split(X
```

```
39 , y, test_size=0.2, random_state=None)
40 model = LogisticRegression()
41 model.fit(X_train, y_train)
42
43
44 prediction_test = model.predict(X_test)
45 print('\n', "Logistic Accuracy: " + str(
   accuracy_score(y_test, prediction_test)))
46 print('\n', 'Intercept:', model.intercept_)
47 print('Co-efficients:', model.coef_)
48
49 err = []
50 err_2 = []
51 kf = KFold(n_splits=10, random_state=None, shuffle=
   True)
52 for train_index, test_index in kf.split(X):
53     X_train, X_test = X[train_index], X[test_index]
54     y_train, y_test = y[train_index], y[test_index]
55     model.fit(X_train, y_train)
56     y_hat = model.predict(X_test)
57     err.append(np.average((y_hat - y_test) ** 2))
58     err_2.append(model.score(X_test, y_test))
59 print('\n', "Average Accuracy: " + str(np.average(
   err_2)))
60
61
62 # Logistic Regression Cross Validation
63 lrcv = linear_model.LogisticRegressionCV(penalty='l1'
   , solver='liblinear', cv=10, max_iter=1500)
64 lrcv.fit(X_train, y_train)
65
66 kf = KFold(n_splits=10, random_state=1234, shuffle=
   True)
67 num_cols = 13
68 offset = 0.000001
69 number_of_steps = 100
70 maxC = 1
71 step = maxC / number_of_steps
72
73 #Creating the range of Lambdas
74
```

```
 75 unique_lambdas = np.arange(0 + offset, maxC + step,
    step)
 76
 77 save_avg_coef = []
 78 save_avg_intercept = []
 79
 80
 81
 82 for the_lambda in unique_lambdas:
 83     sum_coef = [0] * num_cols
 84     sum_intercept = 0
 85     for train_index, test_index in kf.split(X):
 86         X_train, X_test = X[train_index], X[
    test_index]
 87         y_train, y_test = y[train_index], y[
    test_index]
 88         clf = linear_model.LogisticRegression(C=
    the_lambda)
 89         clf.fit(X_train, y_train)
 90         for i in range(len(clf.coef_[0])):
 91             sum_coef[i] += clf.coef_[0][i]
 92         sum_intercept += clf.intercept_
 93
 94     avg_coef = [0] * num_cols
 95     for i in range(len(sum_coef)):
 96         avg_coef[i] = sum_coef[i] / 10
 97     save_avg_coef.append(avg_coef)
 98     print('\n', the_lambda, clf.coef_, clf.
    intercept_)
 99
100 plt.figure(figsize=(12, 6))
101 ax = plt.gca()
102 ax.plot(unique_lambdas, save_avg_coef)
103 ax.set_xscale('log')
104 plt.xlabel('lambda')
105 plt.ylabel('weights')
106 plt.title('Lasso coefficients')
107 plt.tight_layout()
108 plt.show()
109
110
```