PS9_psy

March 19, 2019

1 PS8

1.1 Siyuan Peng

1.2 1

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVC
from scipy.stats import randint as sp_randint
from scipy.stats import uniform as sp_uniform
from sklearn.neural_network import MLPClassifier
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv('data/strongdrink.txt', na_values='?')
df.head(10)
```

Out[2]:

	cultivar	alco	malic	ash	alk	magn	tot_phen	flav	nonfl_phen	\
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
5	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	
6	1	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	
7	1	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31	
8	1	14.83	1.64	2.17	14.0	97	2.80	2.98	0.29	
9	1	13.86	1.35	2.27	16.0	98	2.98	3.15	0.22	

	proanth	color_int	hue	OD280rat	proline
0	2.29	5.64	1.04	3.92	1065
1	1.28	4.38	1.05	3.40	1050
2	2.81	5.68	1.03	3.17	1185
3	2.18	7.80	0.86	3.45	1480
4	1.82	4.32	1.04	2.93	735
5	1.97	6.75	1.05	2.85	1450

```
6      1.98        5.25  1.02       3.58      1290
7      1.25        5.05  1.06       3.58      1295
8      1.98        5.20  1.08       2.85      1045
9      1.85        7.22  1.01       3.55      1045
```

1.3 (a)

```
In [5]: %matplotlib notebook
        for cultivar, group in df.groupby(['cultivar']):
            plt.scatter(group['alco'], group['color_int'],label='cultivar =
        {}'.format(cultivar))
        plt.legend()
        plt.xlabel("alcohol")
        plt.ylabel("color intensity")
        plt.title('Relationship between Alcohol and Color Intensity')
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


Out[5]: Text(0.5, 1.0, 'Relationship between Alcohol and Color Intensity')
```

1.4 (b)

```
In [6]: X = df[['alco','malic','tot_phen','color_int']]
        y = df['cultivar']
        LR = LogisticRegression(random_state=25)
        param_dist1 = {'penalty': ['l1', 'l2'], 'C': sp_uniform(0.1, 10.0)}
        rscv_lr = RandomizedSearchCV(LR, param_dist1,
                    n_iter=200, n_jobs=-1, cv=5, random_state=25,
        scoring='neg_mean_squared_error')


        lr = rscv_lr.fit(X, y)
        print('Optimal tuning parameter values:\n', lr.best_params_)
        print('MSE of the optimal results:', abs(lr.best_score_))
```

```
Optimal tuning parameter values:
 {'C': 2.665871587495725, 'penalty': 'l1'}
MSE of the optimal results: 0.11931818181818182
```

1.5 (c)

```
In [12]: from sklearn.ensemble import RandomForestClassifier

         param_dist2 = {'n_estimators': sp_randint(10, 200),
                        'max_depth': sp_randint(2, 4),
                        'min_samples_split': sp_randint(2, 20),
                        'min_samples_leaf': sp_randint(2, 20),
                        'max_features': sp_randint(1, 4)}
         RFC = RandomForestClassifier(random_state=25)
         rscv_rf = RandomizedSearchCV(RFC, param_dist2,
                  n_iter=200, n_jobs=-1, cv=5, random_state=25, scoring='neg_mean_squared_error')

         rf = rscv_rf.fit(X, y)
         print('Optimal tuning parameter values:\n', rf.best_params_)
         print('MSE of the optimal results:', abs(rf.best_score_))
```

```
Optimal tuning parameter values:
 {'max_depth': 3, 'max_features': 1, 'min_samples_leaf': 13, 'min_samples_split': 18,
'n_estimators': 176}
MSE of the optimal results: 0.13068181818181818
```

1.6 (d)

```
In [13]: param_dist3 = {'C': sp_uniform(loc=0.1, scale=10.0),
                        'gamma': ['scale', 'auto'],
                        'shrinking': [True, False]}
         svc = SVC(kernel='rbf', random_state=25)
         rscv_SVC = RandomizedSearchCV(svc, param_dist3,
                 n_iter=200, n_jobs=-1, cv=5, random_state=25, scoring='neg_mean_squared_error')

         random_SVC = rscv_SVC.fit(X, y)
         print('Optimal tuning parameter values:\n', random_SVC.best_params_)
         print('MSE of the optimal results:', abs(random_SVC.best_score_))
```

```
Optimal tuning parameter values:
 {'C': 3.3605112613782553, 'gamma': 'scale', 'shrinking': True}
MSE of the optimal results: 0.14772727272727273
```

1.7 (e)

```
In [16]: param_dist4 = {'hidden_layer_sizes': sp_randint(1, 100),
                        'activation': ['logistic', 'relu'],
                        'alpha': sp_uniform(0.1, 10.0)}
         mlp = MLPClassifier(activation='tanh', solver='lbfgs', alpha=1, random_state=25)
         rscv_MLP = RandomizedSearchCV(mlp, param_dist4,
                 n_iter=200, n_jobs=-1, cv=5, random_state=25, scoring='neg_mean_squared_error')

         random_MLP = rscv_MLP.fit(X, y)
         print('Optimal tuning parameter values:\n', random_MLP.best_params_)
         print('MSE of the optimal results:', abs(random_MLP.best_score_))
```

```
Optimal tuning parameter values:
 {'activation': 'relu', 'alpha': 0.7965389843643799, 'hidden_layer_sizes': 91}
MSE of the optimal results: 0.06818181818181818
```

1.8 (f)

From my perspective, the MLPClassifier is the best method, considering that it has the lowest MSE value.