Machine Learning Project on Wisconsin Diagnostic Breast Cancer (WDBC) data

인공지능 에이전트 / 기계학습 / 조민철

1. Introduction

WDBC 데이터는 유방의 종양이 악성(malignant)인지 양성(benign)인지 이미 label이 매겨진 데이터이다. 본 리포트에서는 classification 문제에 쓰이는 머신러닝 알고리즘 Logistic Regression, Random Forest, Neural Network를 사용해서 모델을 만들고 cross validation을 통해 각 모델의 정확도를 평가하고 비교했다. 또한 label이 없다고 가정하고 K-means 알고리즘올 clustering 한 후에 실제 label과 얼마나 차이가 있는지 비교하고, cluster의 개수에 따라 score가 어떻게 변하는지 살펴보면서 최적화된 cluster 개수가 실제 데이터의 class 개수와 일치하는지 살펴봤다.

2. Methods

2.1 Logistic Regression

Logistic Regression은 binary classification에서 사용되는 대표적인 모델 중 하나이다. 2개의 class 데이터를 직선으로 구별하는 linear classifier를 만들고, 두 데이터가 어떤 class에 속할지 확률을 이용해서 계산할 수 있다.

2.2 Random Forest

Random Forest는 feature의 일부를 무작위로 선정해서 여러 개의 의사결정 트리를 만든 후에 모든 트리를 살펴보고 다수의 결정을 따르는 모델이다. 의견을 통합하거나 여러 가지 결과를 합치는 방식이므로 앙상블 학습이라고도 한다.

2.3 Neural Network (Multilayer Perceptron)

Neural Network는 input layer와 output layer 사이에 여러 개의 hidden layer를 갖추고 있는 모델이다. activation function으로 non-linear function을 사용해서 layer를 쌓기 때문에 복잡한 데이터도 잘 구분해낼 수 있는 장점이 있지만, hidden layer에서 어떤 일이 발생하는지 사람이 알 수 없다는 단점이 있다.

2.4 K-means clustering

K-means clutering은 위 3가지 알고리즘처럼 classification에서 사용되는 모델이 아니라 clustering에서 사용되는 모델이다. 특정 cluster 내부에 속한 데이터들은 다른 cluster에 속한 데이터들에 비해 더 가까운 거리를 가질 것라는 속성을 활용해서 주어진 데이터를 K개의 cluster로 묶어주는 알고리즘이다. K-means clustering의 결과는 Rand Index와 Silhouette score로 판단한다. Rand Index는 clustering된 각 데이터에 대해, 기준 label과 비교하여 일치하는 수와 불일치하는 수로부터 계산된다. Silhouette score는 그 값이 클수록 서로 다른 cluster 간의 거리가 멀리 떨어져 있음을 알려주는 지표이다.

3. Results and Discussion

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns  # used for plot interactive graph

from sklearn import preprocessing
from sklearn.model_selection import cross_val_score

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn import metrics  # for the check the error and accuracy
 of the model

%matplotlib inline
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv("data/data.csv", header=0)

# Remove unnecessary columns
df.drop('id',axis=1,inplace=True)
df.drop('Unnamed: 32',axis=1,inplace=True)

# Change categorical strings to numerical values
df['diagnosis']=df['diagnosis'].map({'M':1,'B':0})

prediction_var = list(df)[1:]

train_X = df[prediction_var]
train_y = df.diagnosis
```

3.1 Logistic Regression

- 두 class 데이터를 직선으로 경계를 나누는 linear classifier의 각 variable에 대한 weight와 intercept를 포함한 coefficients는 아래와 같았다.

In [3]:
```
model = LogisticRegression()
model.fit(train_X, train_y)

model_scores = cross_val_score(model, train_X, train_y, cv=10, scor
ing='accuracy')
print ('cross_val_score: %f' % model_scores.mean())
```

```
cross_val_score: 0.950900
```

In [4]:
```
coefs = np.append(model.coef_[0], model.intercept_)

coefs_index = list(prediction_var)
coefs_index.append('*intercept')

coef = pd.Series(coefs, index=coefs_index).sort_values(ascending=Fa
lse)
print coef
```

```
concavity_worst            1.571790
compactness_worst          1.118516
symmetry_worst             0.690282
concave points_worst       0.653821
concavity_mean             0.643578
compactness_mean           0.399825
texture_worst              0.346272
concave points_mean        0.340198
smoothness_worst           0.284643
symmetry_mean              0.225914
smoothness_mean            0.153785
perimeter_worst            0.125756
fractal_dimension_worst    0.112852
area_se                    0.094665
perimeter_mean             0.057332
concavity_se               0.046433
symmetry_se                0.042220
concave points_se          0.039987
fractal_dimension_mean     0.026002
radius_se                  0.025958
area_worst                 0.023992
smoothness_se              0.016845
area_mean                  0.003596
perimeter_se              -0.001623
fractal_dimension_se      -0.006408
compactness_se            -0.006984
texture_mean              -0.122052
*intercept                -0.390319
radius_worst              -1.245279
texture_se                -1.248187
radius_mean               -2.101300
dtype: float64
```

3.2 Random Forest

- Random Forest 결과 가장 informative한 상위 5개 feature는 순서대로 **perimeter_worst, concave points_worst, concave points_mean, area_worst, radius_worst**였다. 각 feature의 information strength는 아래 코드의 출력 결과에 나와 있다.

In [5]:
```
model = RandomForestClassifier(n_estimators=100)
model.fit(train_X,train_y)
model_scores = cross_val_score(model, train_X, train_y, cv=10, scor
ing='accuracy')
print ('cross_val_score: %f' % model_scores.mean())
```

```
cross_val_score: 0.961520
```

In [6]:
```
featimp = pd.Series(model.feature_importances_, index=prediction_va
r).sort_values(ascending=False)
print(featimp[:5])
```

```
perimeter_worst         0.133672
area_worst              0.128971
concave points_worst    0.117441
radius_worst            0.104748
concave points_mean     0.094516
dtype: float64
```

3.3 Neural Network (MLP)

- Neural Network 알고리즘은 hyperparameter인 hidden layer의 개수를 바꿔가며 3가지 모델을 만들었다. 기본적으로 hidden layer 하나당 hidden units를 128개로 두고, hidden layer의 개수를 추가하는 식으로 3가지 모델을 구성했다. hidden layer의 개수가 늘어날수록 cross validation score가 증가하는 것을 알 수 있었다. 다만, hidden layer의 수가 증가할 때마다 parameter 개수가 기하급수적으로 늘어나는 만큼 모델이 overfitting 될 확률이 매우 높아진다. 복잡도가 높은 모델은 test data에 대한 generalization 효과가 떨어지기 때문에 정확도에 큰 차이가 없다면 parameter 수가 적은 모델을 사용하는 것이 더 옳다.

In [7]:
```
model = MLPClassifier(hidden_layer_sizes=(128))
model.fit(train_X,train_y)
model_scores = cross_val_score(model, train_X, train_y, cv=10, scor
ing='accuracy')
print ('cross_val_score: %f' % model_scores.mean())
```

```
cross_val_score: 0.792812
```

In [8]:
```
model = MLPClassifier(hidden_layer_sizes=(128,128,128))
model.fit(train_X,train_y)
model_scores = cross_val_score(model, train_X, train_y, cv=10, scor
ing='accuracy')
print ('cross_val_score: %f' % model_scores.mean())
```

```
cross_val_score: 0.901275
```

In [9]:
```
model = MLPClassifier(hidden_layer_sizes=(128,128,128,128,128))
model.fit(train_X,train_y)
model_scores = cross_val_score(model, train_X, train_y, cv=10, scor
ing='accuracy')
print ('cross_val_score: %f' % model_scores.mean())
```

```
cross_val_score: 0.917314
```

3.4 K-means

- Adjusted Rand Index와 Silhouette Score를 기준으로 보면 cluster의 개수 k=2일 때 가장 좋은 결과가 나왔다. 실제 데이터 역시 class가 2개 존재했고 clustering 또한 2개로 나누는 것이 가장 성능이 좋다는 것을 보여줬다.
- 그러나 실제 classification label과 clustering result를 비교해보니 정확도는 85.41%에 그쳤다.

In [10]:
```
from sklearn.cluster import KMeans

df2 = df.iloc[:,1:]
X = df2
y = df.iloc[:,0]

estimators = [('k=2', KMeans(n_clusters=2)),('k=4', KMeans(n_cluste
rs=4)),
              ('k=5', KMeans(n_clusters=5)),('k=7', KMeans(n_cluste
rs=7)),
              ('k=10', KMeans(n_clusters=10))]
```

In [11]:
```
print ("Estimator\tHomogeneity\tCompleteness\tV-means\tARI\tAMI\tSi
lhouette")
titles = ['2 clusters', '4 clusters', '5 clusters', '7 clusters',
'10 clusters']
for name, est in estimators:
    est.fit(X)
    labels = est.labels_
    print('%s\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f' % (name,
        metrics.homogeneity_score(y, est.labels_), metrics.complete
ness_score(y, est.labels_),
        metrics.v_measure_score(y, est.labels_), metrics.adjusted_r
and_score(y, est.labels_),
        metrics.adjusted_mutual_info_score(y, est.labels_),
        metrics.silhouette_score(X, est.labels_,
metric='euclidean')))
```

```
Estimator       Homogeneity     Completeness    V-means ARI     AMI
        Silhouette
k=2     0.422   0.517   0.465   0.491   0.422   0.697
k=4     0.575   0.333   0.422   0.413   0.332   0.533
k=5     0.602   0.298   0.398   0.342   0.296   0.510
k=7     0.626   0.247   0.354   0.260   0.244   0.468
k=10    0.638   0.216   0.323   0.190   0.213   0.455
```

In [12]:
```
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
y_np = y.as_matrix()
print 'accuracy:%f' % metrics.accuracy_score(y_np, kmeans.labels_)
```

```
accuracy:0.854130
```

4. Conclusion

지금까지 4가지 알고리즘을 사용해서 WDBC 데이터에 대한 predictive model을 만들고 모델의 성능을 평가 및 비교했다. 정확도 측면에서 보면, classification에 사용한 3가지 알고리즘 중 cross validation score가 가장 높은 알고리즘은 Random Forest (96.15%) > Logistic Regresson (95.09%) > Neural Network (91.73%) 순이었다. 따라서 test data를 예측하기 위해서는 Random Forest 모델을 사용하는 것이 가장 성능이 높을 것으로 예상된다. 그리고 clustering 알고리즘인 K-means 모델의 결과를 보면 Silhouette score가 가장 높은 모델은 cluster의 개수 k=2일 때였다. 따라서 만약 WDBC 데이터에 label이 없었다 하더라도 악성 종양과 양성 종양의 feature만으로도 cluster가 2개로 잘 구분될 가능성이 높다고 할 수도 있지만, clustering result를 데이터의 실제 label과 비교해보면 85.41%만 일치했기 때문에 악성 종양과 양성 종양의 구분이 label 없이는 명확하지 않다는 것을 알 수 있었다.