Data Analytics | Homework Assignment 4

Jiayu Chen (jic117@pitt.edu)
Huan Chen (huc48@pitt.edu)
Chang Tian (cht97@pitt.edu)

1.Create a classification model of your choice (naive Bayes, logistic regression, Bayesian network, neural networks, etc.)

We implement **logistic regression** model.

Firstly, we import data from .csv file

```
import pandas as pd
DataHouseVotes = pd.read_csv('house-votes-84.csv')
```

We find that the features in this dataset have values of ‘y’, ‘n’ and ’w’,

and the target of ‘republican’ and ‘democrat’.

In order to facilitate calculation, we change {‘y’, ‘n’, ‘w’} into {1, 0, -1}

and change {‘republican’, ‘democrat’} into {0, 1}

```
df=DataHouseVotes
df.replace('n',0, inplace=True)
df.replace('y',1, inplace=True)
df.replace('w', -1,inplace=True)
df.replace('republican', 0, inplace=True)
df.replace('democrat', 1, inplace=True)
X_features=df.iloc[:, range(1,17)]
Y_target=df.iloc[:, 0]
```

Then we build a Logistic Regression model

```
LogRegModel=LogisticRegression()
kf = KFold(n_splits=5, random_state=1, shuffle=True)
```

2. Find overall classification accuracy

We use 5-fold test to fit model. The accuracy is calculated by 5-fold cross-validation.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
accuracy=0
kf = KFold(n_splits=5, random_state=1, shuffle=True)
for train_index, test_index in kf.split(X_features):
        i+=1
        train=train_index.tolist()
        test=test_index.tolist()
        X_train, X_test = X_features.iloc[train], X_features.iloc[test]
        Y_train, Y_test = Y_target.iloc[train_index], Y_target.iloc[test_index]
        fitted_model=LogRegModel.fit(X_train, Y_train)
        pre_y=LogRegModel.predict(X_test)
        accuracy += LogRegModel.score(X_test, Y_test)
r_square_mean=accuracy/5
print('The r square mean is:', r_square_mean)
```

Result:

```
The accuracy score is: 0.947126436782
```

So, **the overall classification accuracy is 0.947126436782**

3. Report sensitivity and specificity for each of the two parties

4. Measure positive and negative predictive value for each of the two parties

These two questions are done based on this function.

```
def cal_sensitivity_specificity(Y_test, pre_y, party1, party2):
        TP=0;FP=0;FN=0;TN=0
        for i in range(Y_test.shape[0]):
                if(Y_test.iloc[i]==party1):
                        if(pre_y[i]==party1):
                                TP+=1
                        elif(pre_y[i]==party2):
                                FN+=1
```

```
                elif(Y_test.iloc[i]==party2):
                        if(pre_y[i]==party1):
                                FP+=1
                        elif(pre_y[i]==party2):
                                TN+=1
        sen = TP/(TP+FN)
        spe = TN/(TN+FP)
        PPV = TP/(TP+FP)
        NPV = TN/(TN+FN)
        return sen, spe, PPV, NPV

sum_sen_de=0; sum_spe_de=0; sum_PPV_de=0; sum_NPV_de=0;
sum_sen_re=0; sum_spe_re=0; sum_PPV_re=0; sum_NPV_re=0;
```

Then calculate the average sensiticity/specifity and positive/negative predictive value for each of the two parties.

Put this code into 5-fold circulation:

```
sen_de, spe_de, PPV_de, NPV_de = cal_sensitivity_specificity(Y_test, pre_y, 1, 0)
sum_sen_de+=sen_de; sum_spe_de+=spe_de; sum_PPV_de+=PPV_de; sum_NPV_de+=NPV_de;
sen_re, spe_re, PPV_re, NPV_re = cal_sensitivity_specificity(Y_test, pre_y, 0, 1)
sum_sen_re+=sen_re; sum_spe_re+=spe_re; sum_PPV_re+=PPV_re; sum_NPV_re+=NPV_re;
```

Calculate the average results:

```
sen_de_mean = sum_sen_de/5
spe_de_mean = sum_spe_de/5
PPV_de_mean = sum_PPV_de/5
NPV_de_mean = sum_NPV_de/5

sen_re_mean = sum_sen_re/5
spe_re_mean = sum_spe_re/5
PPV_re_mean = sum_PPV_re/5
NPV_re_mean = sum_NPV_re/5
```

Result:

```
sensitivity of democrat is: 0.9564716017986141
specificity of democrat is: 0.9378237494156147
```

```
sensitivity of republican is: 0.9378237494156147
specificity of republican is: 0.9564716017986141

positive predictive value of democrat is: 0.9614222391417414
negative predictive value of democrat is: 0.9126024955436719
positive predictive value of republican is: 0.9126024955436719
negative predictive value of republican is: 0.9614222391417414
```

5) Plot ROC curve of your model.

```
from scipy import interp
from sklearn.metrics import roc_curve, auc
tprs = []
mean_fpr = np.linspace(0, 1, 100)
```

Put these code into 5-fold circulation and draw every roc curve:

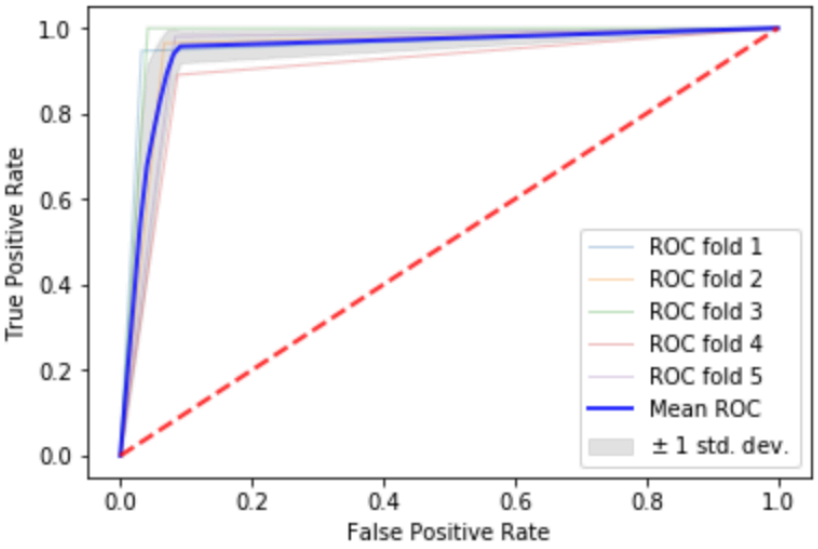
```
fpr, tpr, thresholds = roc_curve(Y_test, pre_y)
tprs.append(interp(mean_fpr, fpr, tpr))
tprs[-1][0] = 0.0
plt.plot(fpr, tpr, lw=1, alpha=0.3,label='ROC fold %d' % (i))
```

Then draw the average roc curve using blue line:

```
mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.plot(mean_fpr, mean_tpr, color='b',label=r'Mean ROC',lw=2, alpha=.8)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2, label=r'$\pm$ 1 std. dev.')

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', alpha=.8)
plt.legend()
plt.show()
```

Result:


True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC fold 1
ROC fold 2
ROC fold 3
ROC fold 4
ROC fold 5
Mean ROC
± 1 std. dev.