Logistic Regression using stratified k-folds cross validation

Bookmarked

To check how robust our model is to unseen data, we can use Validation. It is a technique which involves reserving a particular sample of a dataset on which you do not train the model. Later, you test your model on this sample before finalizing it. Some of the common methods for validation are listed below:

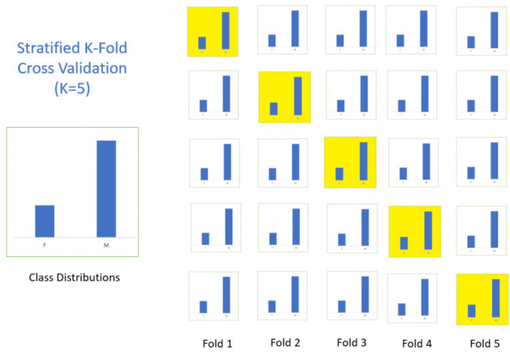
- The validation set approach
- k-fold cross validation
- Leave one out cross validation (LOOCV)
- Stratified k-fold cross validation

If you wish to know more about validation techniques, then please refer this article: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

In this section we will learn about **stratified k-fold cross validation**. Let us understand how it works:

- Stratification is the process of rearranging the data so as to ensure that each fold is a good representative of the whole.
- For example, in a binary classification problem where each class comprises of 50% of the data, it is best to arrange the data such that in every fold, each class comprises of about half the instances.
- It is generally a better approach when dealing with both bias and variance.
- A randomly selected fold might not adequately represent the minor class, particularly in cases where there is a huge class imbalance.

Below is the visualization of a stratified k-fold validation when k=5.



Let's import StratifiedKFold from sklearn and fit the model.

```
from sklearn.model_selection import StratifiedKFold
```

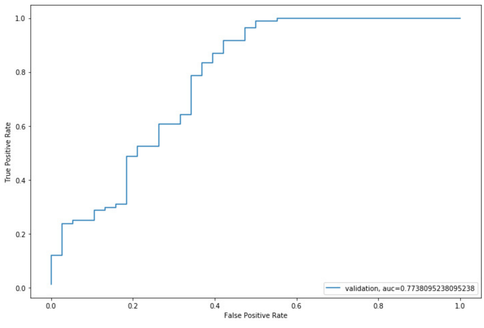
Now let's make a cross validation logistic model with stratified 5 folds and make predictions for test dataset.

```
i=1
kf = StratifiedKFold(n_splits=5,random_state=1,shuffle=True)
for train_index,test_index in kf.split(X,y):
     print('\n{} of kfold {}'.format(i,kf.n_splits))
     xtr,xvl = X.loc[train_index],X.loc[test_index]
     ytr,yvl = y[train_index],y[test_index]

     model = LogisticRegression(random_state=1)
     model.fit(xtr, ytr)
     pred_test = model.predict(xvl)
     score = accuracy_score(yvl,pred_test)
     print('accuracy_score',score)
     i+=1
pred_test = model.predict(test)
pred=model.predict_proba(xvl)[:,1]
```

```
1 of kfold 5
accuracy_score 0.7983870967741935

2 of kfold 5
accuracy_score 0.8306451612903226

3 of kfold 5
accuracy_score 0.8114754098360656

4 of kfold 5
accuracy_score 0.7950819672131147

5 of kfold 5
accuracy_score 0.8278688524590164
```

The mean validation accuracy for this model turns out to be 0.81. Let us visualize the roc curve.

```
from sklearn import metrics
fpr, tpr, _ = metrics.roc_curve(yvl,  pred)
auc = metrics.roc_auc_score(yvl, pred)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="validation, auc="+str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```



We got an auc value of 0.77.

```
submission['Loan_Status']=pred_test
submission['Loan_ID']=test_original['Loan_ID']
```

Remember we need predictions in Y and N. So let's convert 1 and 0 to Y and N.

```
submission['Loan_Status'].replace(0, 'N',inplace=True)
submission['Loan_Status'].replace(1, 'Y',inplace=True)
```

Lets convert the submission to .csv format and make submission to check the accuracy on the leaderboard.

```
pd.DataFrame(submission, columns=['Loan_ID','Loan_Status']).to_csv('Logistic.csv')
```

From this submission we got an accuracy of 0.78472 on the leaderboard. Now we will try to improve this accuracy using different approaches.