Empirical Software Engineering (SE-404)

LAB A1-G2

Laboratory Manual

Department of Software Engineering

DELHI TECHNOLOGICAL UNIVERSITY(DTU)

Shahbad Daulatpur, Bawana Road, Delhi-110042

Submitted to: -

Ms. Priya Singh

Submitted by:-

Name: ASHISH KUMAR

Roll number: 2K18/SE/041

INDEX

S.No.	EXPERIMENT	DATE	REMARKS
10.	Perform a comparison of the following data analysis tools. WEKA, KEEL, SPSS, MATLAB, R.	04-01-2022	
1.	Consider any empirical study of your choice (Experiments, Survey Research, Systematic Review, Postmortem analysis and case study). Identify the following components for an empirical study: a. Identify parametric and nonparametric tests b. Identify Independent, dependent and confounding variables c. Is it Within-company and cross-company analysis? d. What type of dataset is used? Proprietary and open-source software	18-01-2022	
2.	Defect detection activities like reviews and testing help in identifying the defects in the artifacts (deliverables). These defects must be classified into various buckets before carrying out the root cause analysis. Following are some of the defect categories: Logical, User interface, Maintainability, and Standards. In the context of the above defect categories, classify the following statements under the defect categories.	25-01-2022	
3.	Consider any prediction model of your choice. a. Analyze the dataset that is given as a input to the prediction model b. Find out the quartiles for the used dataset c. Analyze the performance of a model using various performance metrics.	25-01-2022	
8.	Why is version control important? How many types of version control systems are there? Demonstrate how version control is used in a proper sequence (stepwise).	01-02-2022	
9.	Demonstrate how Git can be used to perform version control?	01-02-2022	
11.	Validate the results obtained in experiment 3 using 10-cross validation, hold out validation or leave one out cross-validation.	15-02-2022	

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 11

- ASHISH KUMAR
- 2K18/SE/041

Experiment Objective:- Validate the results obtained in experiment 3 using 10-cross validation, hold out validation or leave one out cross-validation.

Introduction:- **Cross-Validation** also referred to **as out of sampling technique** is an essential element of a data science project. It is a resampling procedure used to evaluate machine learning models and access how the model will perform for an independent test dataset.

1. Leave p-out cross-validation: Leave p-out cross-validation (LpOCV) is an exhaustive cross-validation technique, that involves using p-observation as validation data, and remaining data is used to train the model. This is repeated in all ways to cut the original sample on a validation set of *p* observations and a training set.

2. Leave-one-out cross-validation: Leave-one-out cross-validation (LOOCV) is an exhaustive cross-validation technique. It is a category of LpOCV with the case of p=1.

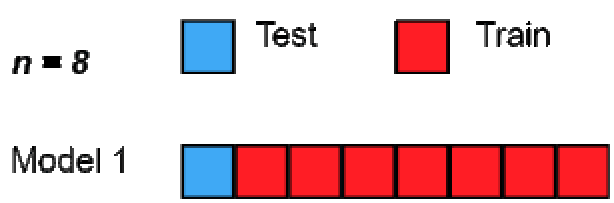

Fig. LOOCV operations

For a dataset having n rows, 1st row is selected for validation, and the rest (n-1) rows are used to train the model. For the next iteration, the 2nd row is selected for validation and rest to train the model. Similarly, the process is repeated until n steps or the desired number of operations.

Both the above two cross-validation techniques are the types of exhaustive cross-validation. Exhaustive cross-validation methods are cross-validation methods that learn and test in all possible ways. They have the same pros and cons discussed below:

Pros:

1. Simple, easy to understand, and implement.

Cons:

1. The model may lead to a low bias.
2. The computation time required is high.

3. Holdout cross-validation: The holdout technique is an exhaustive cross-validation method, that randomly splits the dataset into train and test data depending on data analysis.

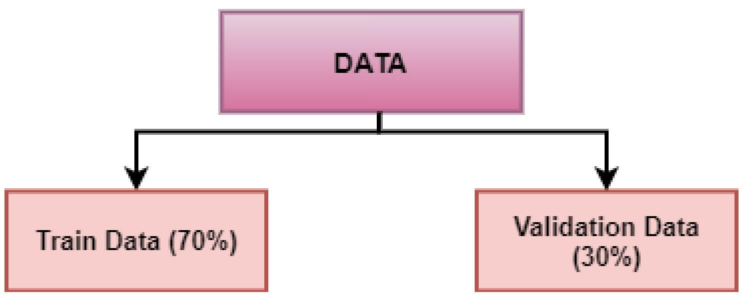

Fig. 70:30 split of Data into training and validation data respectively

In the case of holdout cross-validation, the dataset is randomly split into training and validation data. Generally, the split of training data is more than test data. The training data is used to induce the model and validation data is evaluates the performance of the model.

The more data is used to train the model, the better the model is. For the holdout cross-validation method, a good amount of data is isolated from training.

Pros:

1. Same as previous.

Cons:

1. Not suitable for an imbalanced dataset.
2. A lot of data is isolated from training the model.

4. k-fold cross-validation:

In k-fold cross-validation, the original dataset is equally partitioned into k subparts or folds. Out of the k-folds or groups, for each iteration, one group is selected as validation data, and the remaining (k-1) groups are selected as training data.

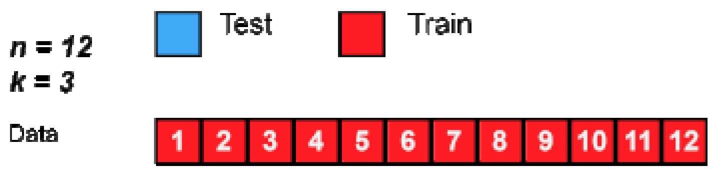

Fig. k-fold cross-validation

The process is repeated for k times until each group is treated as validation and remaining as training data.

The final accuracy of the model is computed by taking the mean accuracy of the k-models validation data.

$$\mathrm{acc}_{\mathrm{cv}} = \sum_{\mathrm{i}=1}^{\mathrm{k}} \frac{\mathrm{acc}_{\mathrm{i}}}{\mathrm{k}}$$

LOOCV is a variant of k-fold cross-validation where k=n.

Pros:

1. The model has low bias
2. Low time complexity
3. The entire dataset is utilized for both training and validation.

Cons:

1. Not suitable for an imbalanced dataset.

Note:- I have used diabetes.csv dataset which contains 768 observations and 9 variables, as described below:

1. pregnancies - Number of times pregnant.
2. glucose - Plasma glucose concentration.
3. diastolic - Diastolic blood pressure (mm Hg).
4. triceps - Skinfold thickness (mm).
5. insulin - Hour serum insulin (mu U/ml).
6. bmi - BMI (weight in kg/height in m).
7. dpf - Diabetes pedigree function.
8. age - Age in years.
9. diabetes - "1" represents the presence of diabetes while "0" represents the absence of it. This is the target variable.

CODE (in python):-

```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn

# Import necessary modules
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import LeavePOut
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
dat = pd.read_csv('diabetes.csv')
print(dat.shape)
dat.describe().transpose()
x1 = dat.drop('class', axis=1).values
y1 = dat['class'].values
# Evaluate using a train and a test set
# Holdout Validation Approach - Train and Test Set Split
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(x1, y1, test_size=0.30,
random_state=100)
model = LogisticRegression()
model.fit(X_train, Y_train)
```

```
result = model.score(X_test, Y_test)
print("Accuracy: %.2f%%" % (result*100.0))

#K-fold Cross-Validation
kfold = model_selection.KFold(n_splits=10, random_state=100)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, x1, y1, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
y_pred=model.predict(X_test)
cm=confusion_matrix(Y_test,y_pred)
print("Confusion matrix is:\n",cm)

#Leave One Out Cross-Validation (LOOCV)
loocv = model_selection.LeaveOneOut()
model_loocv = LogisticRegression()
results_loocv = model_selection.cross_val_score(model_loocv, x1, y1, cv=loocv)
print("Accuracy: %.2f%%" % (results_loocv.mean()*100.0))
```

Output:-

In [13]:

```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn

# Import necessary modules
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import LeavePOut
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
dat = pd.read_csv('diabetes.csv')
print(dat.shape)
dat.describe().transpose()
```

```
(768, 9)
```

Out[13]:

	count	mean	std	min	25%	50%	75%	max
preg	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
plas	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
pres	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
skin	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
insu	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
mass	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
pedi	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00

jupyter expt 11 Last Checkpoint: Last Tuesday at 8:45 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [18]:

```
# Evaluate using a train and a test set
# Holdout Validation Approach - Train and Test Set Split
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(x1, y1, test_size=0.30, random_state=100)
model = LogisticRegression()
model.fit(X_train, Y_train)
result = model.score(X_test, Y_test)
print("Accuracy: %.2f%%" % (result*100.0))
```

```
Accuracy: 74.46%
```

In [19]:

```
#K-fold Cross-Validation
kfold = model_selection.KFold(n_splits=10, random_state=100)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, x1, y1, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
y_pred=model.predict(X_test)
cm=confusion_matrix(Y_test,y_pred)
print("Confusion matrix is:\n",cm)
```

```
Accuracy: 76.95%
Confusion matrix is:
 [[129  21]
 [ 38  43]]
```

In [21]:

```
#Leave One Out Cross-Validation (LOOCV)
loocv = model_selection.LeaveOneOut()
model_loocv = LogisticRegression()
results_loocv = model_selection.cross_val_score(model_loocv, x1, y1, cv=loocv)
print("Accuracy: %.2f%%" % (results_loocv.mean()*100.0))
```

```
Accuracy: 76.95%
```

Learning from experiment:- We have successfully learned about 10-cross validation, hold out validation, leave one out cross-validation techniques and its pros and cons. We have also successful in using this techniques to analyze a given dataset.