Advance Software Engineering (SE-406)

LAB A1-G3

Laboratory Manual

Department of Software Engineering

DELHI TECHNOLOGICAL UNIVERSITY(DTU)

Shahbad Daulatpur, Bawana Road, Delhi-110042

Submitted to: -

Ms. Parul Sharma

Submitted by:-

Name: ASHISH KUMAR

Roll number: 2K18/SE/041

EXPERIMENT 8

- ASHISH KUMAR
- 2K18/SE/041

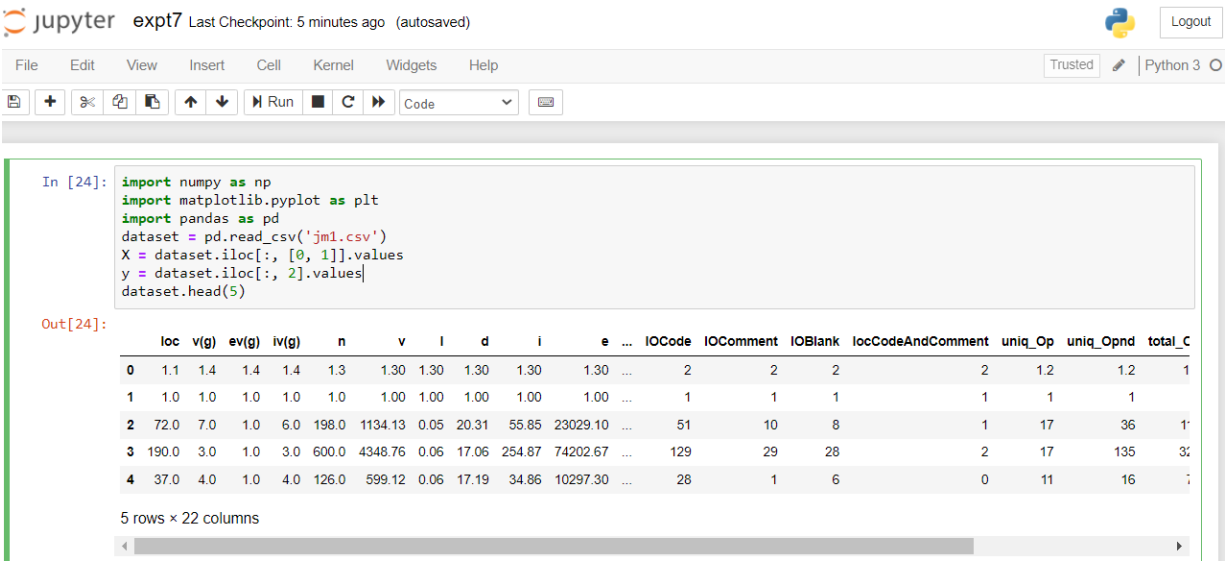
Aim:- To train and test fault prediction model using SVM.

Introduction:- A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression. SVMs are known as maximum margin classifiers as they find the best separating hyperplane between two classes. This process can also be applied recursively to allow the separation of any number of classes. Only those data points that are located nearest to this dividing hyperplane, known as the support vectors, are used by the classifier. This enables SVMs to be used successfully with both large and small data sets.

The aim of this experiment is to observe the classification performance of the Support Vector Machine (SVM) for fault prediction in the context of data sets from the NASA Metrics Data Program (MDP) repository; a collection of data sets generated from NASA software systems and intended for fault prediction research.

Code & Output:-

```
# Importing the dataset
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('jm1.csv')
X = dataset.iloc[:, [0, 1]].values
y = dataset.iloc[:, 2].values
dataset.head(5)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```




```
# Create Training and Test Sets and Apply Scaling
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


#Applying SVM Classifier on the Training Set

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

from sklearn import svm

from sklearn.preprocessing import StandardScaler

from sklearn.svm import LinearSVC, SVC

svm= SVC()

svm.fit(X_train, y_train)

svm_pred = svm.predict(X_test)
```

```
24]:
#svc_model.fit(x_train,y_train)
svm.fit(X_train, y_train)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
24]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Model Accuracy: how often is the classifier correct?

from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(y_test, svm_pred))

```
:
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, svc_pred))
```

```
Accuracy: 0.8467614533965245
```

```
+ Code  + Markdown
```

#visualize support-vectors of svm classifier

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y_train = le.fit_transform(y_train)

from sklearn.svm import SVC

classifier = SVC(kernel='rbf', random_state = 1)

classifier.fit(x_train,y_train)

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(y_train)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/label.py:111: DataConversionWarning: A column-vector y was passed when a 1d array was e
xpected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state = 1)
classifier.fit(x_train,y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=1, shrinking=True,
  tol=0.001, verbose=False)
```

#Visualizing SVM's support vectors

Get support vector indices

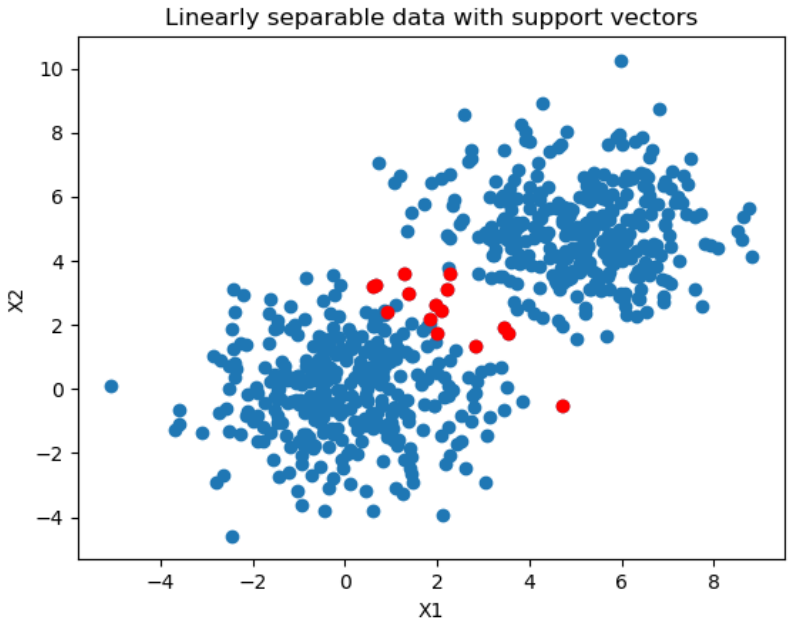
support_vector_indices = classifier.support_

print(support_vector_indices)

```
# Get support vector indices
support_vector_indices = classifier.support_
print(support_vector_indices)
```

```
[   0    1    6    8   10   11   17   24   25   26   28   35   36   39
   53   54   60   61   62   63   64   66   74   76   78   79   80   81
   84  100  104  109  113  117  118  119  122  125  126  128  130  134
  135  137  138  139  140  142  151  152  153  158  162  164  167  175
  177  178  179  183  187  191  194  195  201  209  210  211  212  215
  224  226  227  228  232  236  237  239  241  243  244  248  249  250
  251  252  253  254  260  261  262  266  270  271  273  275  276  277
  278  279  289  290  291  299  307  311  314  315  316  320  327  328
  329  332  333  337  339  341  345  346  351  354  357  360  363  370
  371  375  376  382  383  386  392  396  397  398  399  400  402  408
  410  412  414  415  417  419  422  423  424  426  429  433  435  437
  438  440  441  442  443  444  445  446  447  449  450  459  460  464
  471  473  476  489  498  499  503  506  508  514  520  522  526  528
  529  530  534  535  539  540  541  542  547  553  556  558  561  572
  573  575  576  579  580  582  583  587  588  589  591  592  593  598
  599  602  603  604  607  612  618  620  627  630  635  637  638  642
  644  645  646  648  649  655  656  658  659  660  661  662  666  667
  669  670  671  674  677  681  683  686  687  688  689  691  693  696
  697  702  705  707  712  714  717  724  725  726  727  728  729  734
  739  740  742  745  746  748  750  756  759  760  765  771  774  775
  779  783  784  785  788  790  791  792  794  795  796  798  801  802
  810  811  814  815  818  819  820  822  833  838  839  840  848  849
  850  856  859  864  866  867  869  871  873  874  877  879  880  881
  886  888  890  892  893  894  897  898  899  900  901  904  908  911
  913  914  916  918  919  926  928  929  932  933  936  939  942  943
  945  948  949  950  951  952  953  954  959  960  961  962  967  970
  971  973  977  978  979  981  985  987  989  993  994  997  999 1002
 1004 1006 1008 1009 1011 1012 1013 1015 1019 1020 1021 1022 1028 1029
 1030 1032 1035 1036 1040 1041 1042 1045 1046 1048 1050 1052 1057 1058
 1059 1062 1063 1064 1065 1069 1071 1072 1073 1074 1075 1076 1077 1078
```

```
# Get support vectors themselves

support_vectors = classifier.support_vectors_


# Visualize support vectors

plt.scatter(x_train[:,0].values, x_train[:,1].values)

plt.scatter(support_vectors[:,0].values, support_vectors[:,1].values, color='red')

plt.title('Linearly separable data with support vectors')

plt.xlabel('X1')

plt.ylabel('X2')

plt.show()
```



Result:- The accuracy of the SVM Classifier comes out to be 84.67%.

Learning from experiment:- We have successfully been able to train and test fault prediction model using SVM.