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 7

- ASHISH KUMAR
- 2K18/SE/041

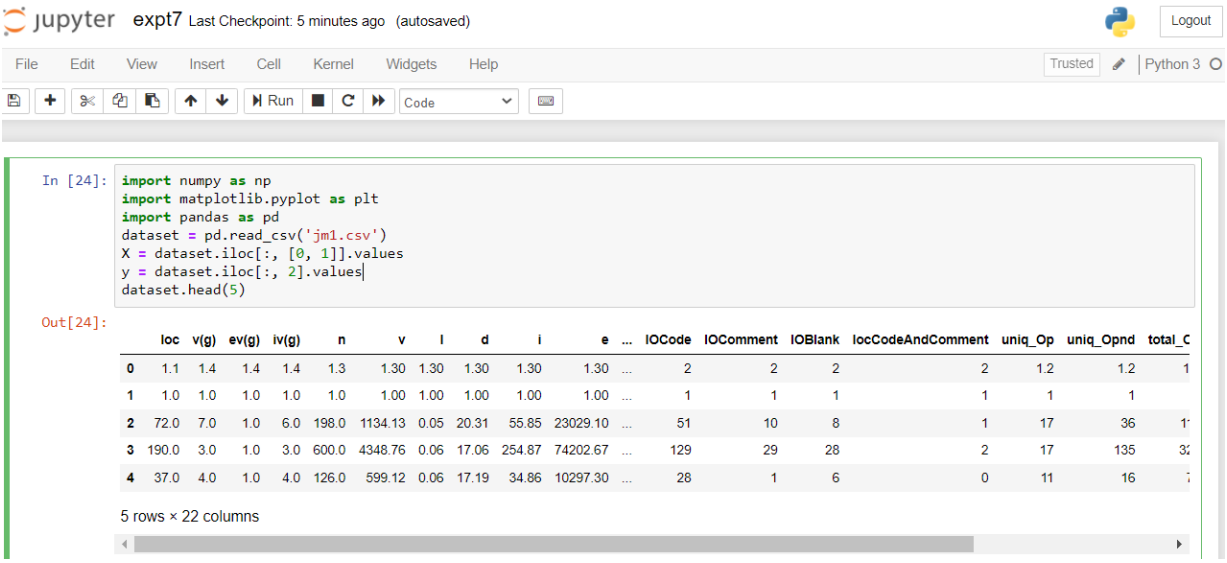
Aim:- Training a support vector machine classifier to predict defect in the dataset collected in experiment 5.

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 defect 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 defect prediction research.

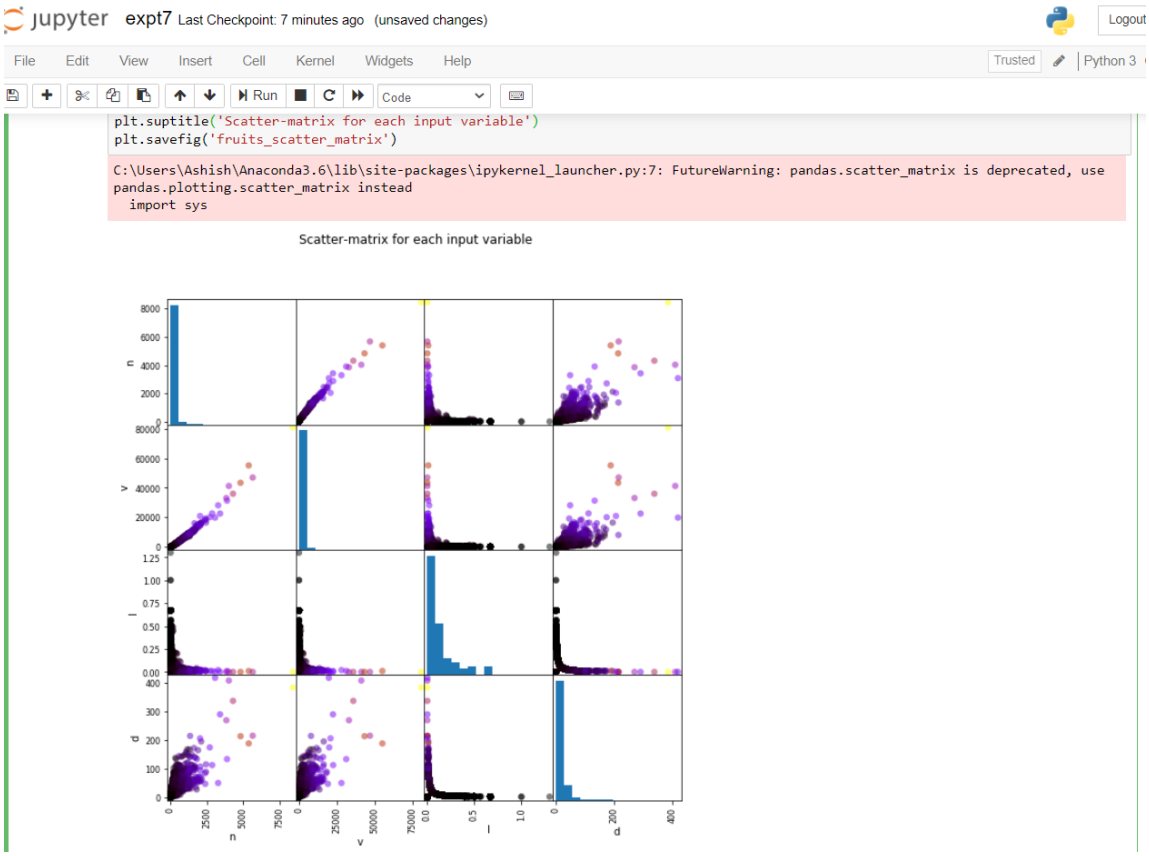
Code & Output:-

```
# 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)
```

```
# Scatter-matrix for each input variable
from pandas.tools.plotting import scatter_matrix
from matplotlib import cm
feature_names = ['n','v','l','d']
X = dataset[feature_names]
y = dataset['loc']
cmap = cm.get_cmap('gnuplot')
scatter = pd.scatter_matrix(X, c = y, marker = 'o', s=40, hist_kwds={'bins':15}, figsize=(9,9),
cmap = cmap)
plt.suptitle('Scatter-matrix for each input variable')
```



#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

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

Learning from experiment:- We have successfully been able to build a **SVM Classification** Model that is able to predict defect in the dataset.